ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

**Διπλωματική Εργασία**

# *Δημιουργία αλγορίθμων και γραφικού περιβάλλοντος για προσδιορισμό κι εμφάνιση γραμματικών στοιχείων σε προτάσεις της ελληνικής γλώσσας*

***Ονοματεπώνυμο: Κωνσταντίνος Αγγέλου***

***Αρ. Μητρώου: 19389077***

***Επιβλέπων:***
***Ευάγγελος Παπακίτσος***

**ΑΙΓΑΛΕΩ 2024**

**UNIVERSITY OF WEST ATTICA**

**SCHOOL OF ENGINEERING**

**DEPARTMENT OF INDUSTRIAL DESIGN AND PRODUCTION ENGINEERING**

**Diploma Thesis**

# Creating algorithms and graphical environment for identification and display of grammatical elements in sentences of the Greek language

**Student name and surname: Konstantinos Angelou**

**Registration Number: 19389077**

**Supervisor name and surname:**

**Evangelos Papakitsos**

**Athens, Year 2024**

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ**
**ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ**
**ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ**

**Δημιουργία αλγορίθμων και γραφικού περιβάλλοντος για προσδιορισμό κι εμφάνιση γραμματικών στοιχείων σε προτάσεις της ελληνικής γλώσσας**

**Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή**

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

| Α/α | ΟΝΟΜΑ ΕΠΩΝΥΜΟ | ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ | ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ |
|---|---|---|---|
| 1 | Ε.Χ. ΠΑΠΑΚΙΤΣΟΣ | ΕΔΙΠ Α΄ | Evangelos Papakitsos<br>Digitally signed by Evangelos Papakitsos<br>Date: 2024.10.23 22:29:51 +03'00' |
| 2 | Ν. ΛΑΣΚΑΡΗΣ | ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ | Nikolaos Laskaris<br>Digitally signed by Nikolaos Laskaris<br>Date: 2024.10.24 13:16:29 +03'00' |
| 3 | Χ. ΔΡΟΣΟΣ | ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ | Digitally signed by CHRISTOS DROSOS<br>Date: 2024.10.25 15:05:58 +03'00' |

## ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος **Κωνσταντίνος Αγγέλου** του **Ελευθερίου**, με αριθμό μητρώου **19389077,** φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής **Μηχανικών** του Τμήματος **Βιομηχανικής Σχεδίασης και Παραγωγής**, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Κωνσταντίνος
Αγγέλου

# ΕΥΧΑΡΙΣΤΙΕΣ

Με το πέρας της διπλωματικής εργασίας θέλω να ευχαριστήσω θερμά τους επιβλέποντες καθηγητές μου. Την κα. Σαμαρίδη και τον κ. Χορόζογλου του Τμήματος Μηχανικών Πληροφορικής & Υπολογιστών της Σχολής Μηχανικών, για τις πολύτιμες συμβουλές και την καθοδήγηση τους για την υλοποίηση της εργασίας. Επίσης, ένα μεγάλο ευχαριστώ στον κ. Παπακίτσο, διότι χωρίς την βοήθεια του δεν θα είχα τελειώσει τόσο σύντομα την διπλωματική μου εργασία.

Τέλος, η διπλωματική αυτή εργασία είναι αφιερωμένη στην οικογένεια μου που με βοήθησε και με στήριξε να φτάσω στο τέλος αυτού του εκπαιδευτικού ταξιδιού.

# Περίληψη

Η ελληνική γλώσσα είναι πάρα πολύ μεγάλη. Αν κάποιος ήθελε να ψάξει ή ακόμα και επεξεργαστεί κάποιες λέξεις, θα έπρεπε να ψάχνει ώρες σε λεξικά, αν δεν είναι γνώστης της ελληνικής γλώσσας. Στην παρούσα διπλωματική εργασία, έχει αναπτυχθεί ένα μοντέλο αλγορίθμων και γραφικού περιβάλλοντος, ως κλάδου της τεχνητής νοημοσύνης, το οποίο θα επεξεργάζεται την ελληνική γλώσσα και θα κάνει εξόρυξη δεδομένων. Αρχικά δίνεται ένα λεξικό από το οποίο φτιάχνεται η βάση δεδομένων. Η βάση δεδομένων έχει φτιαχτεί με τέτοιο τρόπο έτσι ώστε να αποσπά εύκολα πληροφορίες το πρόγραμμα, χωρίς μεγάλη δυσκολία. Ύστερα η εφαρμογή χωρίζει την πρόταση σε λέξεις, ανάλογα με τους κενούς χαρακτήρες και τα σημεία στίξης και θα αναζητάει τις πληροφορίες της κάθε λέξης στη βάση δεδομένων. Οι πληροφορίες που θα παρέχει είναι πολυπληθείς, καθώς επιστρέφει μέρος του λόγου, πτώση, βαθμό, αριθμό, γένος και πολλά άλλα. Επιπλέον μετά την αναζήτηση των λέξεων ξεχωριστά αναγνωρίζει σε κάθε πρόταση το υποκείμενο, ρήμα, άμεσο κι έμμεσο αντικείμενο αυτόματα, με βάση τους κανόνες της γραμματικής. Όταν εξακριβώνει τι είναι η κάθε λέξη, το απαριθμεί και το εκτυπώνει (Relational Grammar). Τέλος η παρούσα διπλωματική συμβάλλει στον αποτελεσματικό και γρήγορο τρόπο εξόρυξης δεδομένων με τη βοήθεια της τεχνητής νοημοσύνης και παρέχει βαθιά εκμάθηση της πλούσιας ελληνικής γλώσσας.

**Λέξεις-κλειδιά**

Εξόρυξη δεδομένων, εύκολα, πολυπληθείς, αυτόματα, βαθιά εκμάθηση.

## Abstract

The Greek language is significantly large. If someone wanted to search for or even process certain words, they would have to spend hours looking through dictionaries if they are not familiar with the Greek language. In this thesis, an algorithmic model and graphical interface will be developed, as a branch of artificial intelligence, which will process the Greek language and perform data mining. Initially, a dictionary will be provided, from which the database will be created for the data mining process. The database will be constructed in such a way that the program can easily extract information without much difficulty. Then, the application will break down the sentence into words, based on the spaces and punctuation marks, and will search for the information of each word in the database. The information provided will be abundant, as it will return the part of speech, case, degree, number, gender, and much more. Additionally, after searching for the words individually, it will automatically recognize the subject, verb, direct and indirect object in each sentence, based on grammar rules. Once it identifies what each word is, it will enumerate and print it (Relational Grammar). Finally, this thesis contributes to an effective and fast way of data mining with the help of artificial intelligence and provides a deep learning experience of the rich Greek language.

## Keywords

Data mining, Easily, Abundant, Automatically, Deep learning

# ΠΕΡΙΕΧΟΜΕΝΑ

# 1. ΕΙΣΑΓΩΓΗ

Η τεχνητή νοημοσύνη (Artificial Intelligence) είναι ένας τομέας της Πληροφορικής που ασχολείται με τη δημιουργία επίλυσης προβλημάτων, ικανών για να εκτελέσουν οποιαδήποτε διεργασία, που υπό κανονικές συνθήκες χρειάζονται ανθρώπινη νοημοσύνη. Αυτές οι διεργασίες μπορεί να είναι η μάθηση, η επίλυση προβλημάτων, η ακόμη και η κατανόηση της φυσικής γλώσσας. Η τεχνητή νοημοσύνη βρίσκεται παντού στις μέρες μας και συνδυάζει στοιχεία από διάφορα πεδία γνώσεων. Μερικά από αυτά, μπορεί να είναι μια άσκηση στα μαθηματικά, μερικές γρήγορες πληροφορίες, οτιδήποτε που μπορεί να διευκολύνει την ζωή του ανθρώπου.

(Πηγή: https://bigblue.academy/gr/ti-einai-i-texniti-noimosuni)

## 1.1 Ιστορική Εξέλιξη

Η πρώτη ιδέα ότι οι μηχανές μπορούν να σκέφτονται και να δρουν σαν τον άνθρωπο ξεκίνησε αρκετά χρόνια πίσω. Ειδικότερα κατά την αρχαιότητα είχαν ειπωθεί πολλές τέτοιες θεωρίες μέσα από μύθους και ιστορίες, όπως ο Τάλως που προστάτευε την Κρήτη. Ωστόσο, η σύγχρονη μελέτη της τεχνητής νοημοσύνης ξεκίνησε από το 1956 με το συνέδριο στο Dartmouth College, όπου ερευνητές, όπως ο John McCarthy, ο Marvin Minsky, και ο Allen Newell, έθεσαν τις βάσεις για τον τομέα της τεχνητής νοημοσύνης.

Σε πρώτο στάδιο οι ερευνητές προσπάθησαν να δημιουργήσουν συστήματα που μιμούνται τους ανθρώπους. Η μίμηση αυτή έχει να κάνει με την ανθρώπινη σκέψη, δηλαδή να δημιουργήσουν συστήματα τα οποία να σκέφτονται σαν τον άνθρωπο. Καθώς προσπαθούσαν να μιμηθούν τον άνθρωπο, αντιμετώπισαν κάποια πρακτικά προβλήματα. Τα προβλήματα αυτά ήταν ότι δεν είχαν τις πληροφορίες για να αναπαριστήσουν τον άνθρωπο. Η ανθρώπινη σκέψη δεν είναι ένα απλό πράγμα, αντιθέτως είναι ένα πολύπλοκο ζήτημα που δεν μπορεί να αναλυθεί εύκολα. Έτσι οι

αναλυτές αναγκάστηκαν να παρατήσουν το έργο τους, μέχρι και σήμερα. Σήμερα με την παρουσία του διαδικτύου η αναζήτηση και συλλογή πληροφοριών είναι πολύ πιο εύκολη από τότε.

Η τεχνητή νοημοσύνη χωρίζεται σε διάφορες υποκατηγορίες, όπου η καθεμία επικεντρώνεται σε συγκεκριμένες πτυχές, όπως:

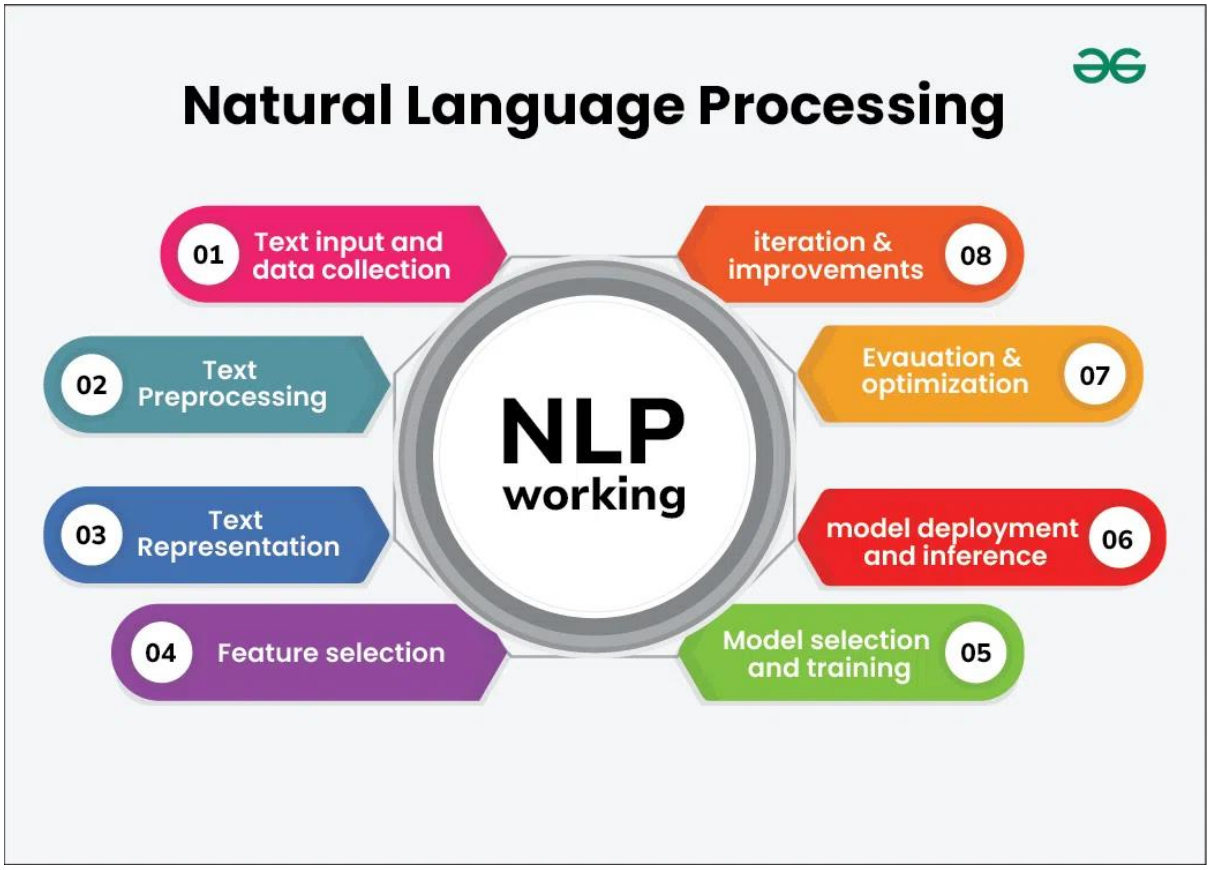
- **Μηχανική Μάθηση (Machine Learning)**: Ασχολείται με τη δημιουργία αλγορίθμων, που επιτρέπουν στα συστήματα να "μαθαίνουν" από τις πληροφορίες. Τα συστήματα δεν είναι μόνο για να εκτελούν συγκεκριμένες εργασίες, αντιθέτως είναι ικανά για να προσαρμόζονται και να βελτιώνονται συνεχώς, καθώς επεξεργάζονται τις νέες πληροφορίες. Η μηχανική μάθηση χρησιμοποιείται σε εφαρμογές όπως αναγνώριση εικόνας, ανάλυση φωνής και συστάσεις προϊόντων.

- **Βαθιά Μάθηση (Deep Learning)**: Πρόκειται για μία μορφή μηχανικής μάθησης που χρησιμοποιεί νευρωνικά δίκτυα τα οποία είναι εμπνευσμένα από τον ανθρώπινο εγκέφαλο. Τα νευρωνικά δίκτυα είναι ικανά να αναγνωρίζουν σύνθετα πρότυπα σε μεγάλες ποσότητες δεδομένων. Από τη βαθιά μάθηση έχει γίνει σημαντική πρόοδος σε τομείς όπως η επεξεργασία φυσικής γλώσσας και η αναγνώριση της εικόνας.

- **Νευρωνικά Δίκτυα (Neural Networks)**: Είναι συστήματα που μιμούνται τη δομή και τη λειτουργία του ανθρώπινου εγκεφάλου και αποτελούνται από έναν αριθμό "νευρώνων" (μονάδες επεξεργασίας). Οι νευρώνες είναι μεταξύ τους συνδεδεμένοι σε επίπεδα και κάθε νευρώνας λαμβάνει εισόδους, τις οποίες επεξεργάζεται και παράγει ως ένα αποτέλεσμα. Το αποτέλεσμα αυτό μπορεί να χρησιμοποιηθεί ως είσοδος για άλλους νευρώνες στο δίκτυο. Αυτή η συνεχής διαδικασία επιτρέπει στα δίκτυα να μαθαίνουν και να αναγνωρίζουν πρότυπα, τα οποία μπορεί να είναι είτε οπτικά είτε ηχητικά ή οτιδήποτε άλλο.

- **Υπολογιστική Όραση (Computer Vision)**: Επικεντρώνεται στην αναγνώριση και κατανόηση οπτικών δεδομένων από τον υπολογιστή και περιλαμβάνει την αναγνώριση αντικειμένων, προσώπων, κίνησης σε εικόνες ή βίντεο. Για παράδειγμα χρησιμοποιείται σε εφαρμογές όπως η αυτόνομη οδήγηση, η ιατρική απεικόνιση, και η ανάλυση βίντεο.

- **Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing - NLP)**: Ασχολείται με την αλληλεπίδραση μεταξύ των υπολογιστών και της ανθρώπινης γλώσσας. Η επεξεργασία φυσικής γλώσσας κατανοεί την ερμηνεία και τη δημιουργία κειμένου ή ομιλίας. Επιπλέον επιτρέπει στους υπολογιστές να εκτελούν εργασίες όπως η μετάφραση της γλώσσας, η ανάλυση συναισθημάτων, και η δημιουργία φυσικών απαντήσεων σε ανθρώπινες ερωτήσεις.



(Πηγή: https://www.geeksforgeeks.org/natural-language-processing-overview/)

## 1.2 Relational Grammar

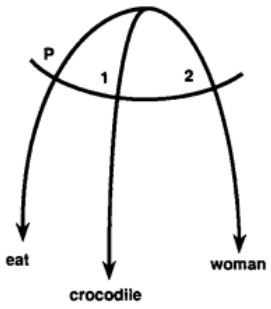
Οι γραμματικές σχέσεις θεωρούνται ως πρωτογενή στοιχεία, τα οποία περιλαμβάνουν το υποκείμενο της πρότασης, το άμεσο αντικείμενο του ρήματος, το έμμεσο αντικείμενο και έναν αδιευκρίνιστο αριθμό πλάγιων σχέσεων. Οι τρείς σχέσεις υποκείμενο, αντικείμενο κι έμμεσο αντικείμενο ονομάζονται συλλογικοί όροι. Αυτές και οι πλάγιες σχέσεις σχηματίζουν μια ιεραρχία, όπως φαίνεται στο παρακάτω σχήμα:

| 1 | 2 | 3 | |
|---|---|---|---|
| υποκείμενο | άμεσο αντικείμενο | έμμεσο αντικείμενο | πλάγιες σχέσεις |

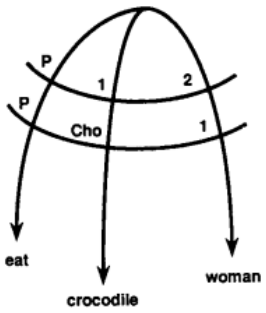
Οι αριθμοί καθορίζουν την ιεραρχία, οπότε το υποκείμενο αναφέρεται ως 1, το άμεσο αντικείμενο ως 2, και το έμμεσο αντικείμενο ως 3. Οι σχέσεις 1 και 2 είναι γνωστές συλλογικά ως πυρηνικές σχέσεις, ενώ οι 2 και 3 ως σχέσεις αντικειμένου.

Για την καλύτερη κατανόηση αναλύεται στο παρακάτω παράδειγμα:

[3a]




[3b]




(Πηγή: Blake, 1990).

The crocodile ate the woman. [3a]
The woman was eaten by the crocodile. [3b]

Η πρόταση θεωρείται ότι έχει ένα ρήμα, το οποίο συμβολίζεται ως P, ένα υποκείμενο, ως 1, και ένα άμεσο αντικείμενο, συμβολίζοντας το ως 2. Στο σχήμα υπάρχει μόνο ένα στρώμα. Στη σχεσιακή δομή του παθητικού παραδείγματος [3b] υπάρχουν δύο στρώματα. Το αρχικό στρώμα παραμένει ίδιο με του ενεργητικού, αλλά το δεύτερο και τελικό στρώμα αντικατοπτρίζει τις αλλαγές. Το αρχικό άμεσο αντικείμενο έχει γίνει ως υποκείμενο και το αρχικό υποκείμενο έχει "υποβιβαστεί" σε "chômeur" (συντομογραφημένο ως Cho).

Η έννοια του "chômeur" είναι σημαντική στη Relational Grammar και αποτελεί μία από τις σημαντικότερες θεωρίες. Όταν μια πρόταση υποστεί έναν συντακτικό μετασχηματισμό, όπως η μετατροπή από ενεργητική σε παθητική φωνή, το αρχικό υποκείμενο μπορεί να χάσει τη θέση του. Σε αυτή την περίπτωση, μετατρέπεται σε "chômeur". Δηλαδή, δεν κατέχει πλέον τη θέση του υποκειμένου, αλλά δεν είναι ούτε πλήρως διαγραμμένο από την πρόταση. Παραμένει μέρος της πρότασης, αλλά σε μια λιγότερο κεντρική ή "υποβαθμισμένη" θέση.

## 1.3 Ιστορικό εφαρμογής

Η Relational Grammar είναι μια θεωρητική προσέγγιση στη γλωσσολογία που αναπτύχθηκε κυρίως τη δεκαετία του `70 από τους David Perlmutter και Paul Postal. Αυτή η θεωρία αποτελεί μέρος της ευρύτερης προσπάθειας για την κατανόηση της συντακτικής δομής των γλωσσών και των γραμματικών σχέσεων μεταξύ των διαφόρων στοιχείων μιας πρότασης.

- **Αρχική Ανάπτυξη (1970):** Η Relational Grammar εμφανίστηκε ως απάντηση σε ορισμένους περιορισμούς που παρουσίαζαν οι τότε κυρίαρχες συντακτικές θεωρίες, όπως η Μετασχηματιστική-Γενετική γραμματική (Transformational-Generative Grammar) που προτάθηκε από τον Noam Chomsky. Η θεωρία του Chomsky επικεντρώθηκε στην σημασία των γραμματικών σχέσεων (όπως το υποκείμενο, το αντικείμενο), καθώς χρησιμοποιήθηκαν ως κεντρικά στοιχεία για την κατανόηση της συντακτικής δομής. Αυτές οι σχέσεις θεωρήθηκαν ως βασικά και ακαθόριστα στοιχεία.

- **Εξέλιξη και Επιρροή (1980):** Η Relational Grammar βοήθησε στη μελέτη των γραμματικών φαινομένων σε πολλές γλώσσες. Μέσω αυτής παράχθηκε ένα πλαίσιο για τη σύγκριση και την ανάλυση τους. Στις αρχές δεν κατέληξε να γίνει η κυρίαρχη θεωρία στη γλωσσολογία, παρόλα αυτά επηρέασε άλλες θεωρητικές προσεγγίσεις, όπως η Lexical-Functional Grammar (LFG) και η Role and Reference Grammar (RRG).

- **Σύγχρονες Εξελίξεις:** Αν και η Relational Grammar δεν βρίσκεται πλέον στο επίκεντρο της γλωσσολογικής έρευνας, παραμένει σημαντική ως μια ιστορική προσέγγιση που έδωσε έμφαση στις γραμματικές σχέσεις. Οι αρχές τις έχουν επηρεάσει τη μελέτη της συντακτικής δομής και κατ' επέκταση έχουν χρησιμοποιηθεί στην ανάπτυξη μοντέλων σε υπολογιστική γλωσσολογία αλλά και τεχνητή νοημοσύνη.

Η Relational Grammar, είναι η μοναδική θεωρία που εστίασε στις γραμματικές σχέσεις, προσφέροντας μια καινοτόμα οπτική στη γλωσσολογική ανάλυση. Έτσι συνέβαλλε στην κατανόηση του πώς οι συντακτικές δομές λειτουργούν και μετασχηματίζονται σε διαφορετικές γλώσσες.

# 2. ΛΕΙΤΟΥΡΓΙΚΕΣ ΠΡΟΔΙΑΓΡΑΦΕΣ

## 2.1 Relational grammar

Μία από τις σημαντικότερες θεωρητικές προσεγγίσεις στη γλωσσολογία και ιδιαίτερα στη συντακτική ανάλυση είναι η Relational Grammar. Η προσέγγιση αυτή δεν έχει την ίδια επίδραση σήμερα όπως άλλες γλωσσολογικές θεωρίες. Παρόλα αυτά έπαιξε κρίσιμο ρόλο στην ανάπτυξη της κατανόησης των γραμματικών σχέσεων και των μετασχηματισμών στις γλώσσες.

## 2.2 Γραμματικές Σχέσεις (Grammatical Relations)

Σε μια πρόταση, η Relational Grammar επικεντρώνεται στις γραμματικές σχέσεις μεταξύ των συντακτικών στοιχείων. Οι σχέσεις ορίζονται τυπικά ως 1, 2 και 3, με το '1' να συμβολίζει το υποκείμενο, το '2' ως άμεσο αντικείμενο και το '3' αντικατοπτρίζει το έμμεσο αντικείμενο. Στις άλλες θεωρίες δίνεται έμφαση σε συντακτικές δομές ή φράσεις, ενώ η Relational Grammar εστιάζει στο πώς αυτά τα στοιχεία σχετίζονται μεταξύ τους.

- **Υποκείμενο (Subject):** Σε μία πρόταση το '1' αντιπροσωπεύει το στοιχείο που εκτελεί την ενέργεια του ρήματος.
- **Άμεσο Αντικείμενο (Direct Object)**: Ο ρόλος του '2', αντιπροσωπεύει το στοιχείο που δέχεται την ενέργεια του ρήματος.
- **Έμμεσο Αντικείμενο (Indirect Object):** Το '3', αφορά το στοιχείο που επηρεάζεται έμμεσα από την ενέργεια του ρήματος.

## 2.3 Στάδια και Διαδοχή Σχέσεων (Strata and Successive Relations)

Η Relational Grammar εισάγει την έννοια των 'στρωμάτων' (strata). Η έννοια αυτή αναπαριστά τα διαφορετικά στάδια μέσω των οποίων μια πρόταση μετασχηματίζεται. Κάθε στρώμα αντικατοπτρίζει ένα συγκεκριμένο στάδιο, όπου οι γραμματικές σχέσεις των στοιχείων μπορεί να αλλάξουν. Για παράδειγμα, μία πρόταση όπως "Ο Γιάννης έδωσε το βιβλίο στη Μαρία" μπορεί να μετασχηματιστεί ως 1 "Ο Γιάννης", 2 ως "το βιβλίο" και 3 "στη Μαρία". Εάν η Πρόταση γίνει σε παθητική φωνή, όπως "Το βιβλίο δόθηκε στη Μαρία από τον Γιάννη" τότε στο επόμενο στρώμα, το βιβλίο προάγεται στο ρόλο του υποκειμένου, ενώ η Μαρία παραμένει ως έμμεσο αντικείμενο.

## 2.4 Προαγωγές και Υποβιβασμοί (Promotions and Demotions)

Ένα άλλο κύριο χαρακτηριστικό της Relational Grammar είναι η εναλλαγή των γραμματικών σχέσεων ως "προαγωγές" (promotions) ή "υποβιβασμοί" (demotions). Η

προαγωγή είναι όταν ένα στοιχείο μίας πρότασης αλλάζει γραμματική σχέση σε ένα επόμενο στρώμα, παίρνοντας σημαντικότερο ρόλο. Για παράδειγμα, όταν το άμεσο αντικείμενο σε μια ενεργητική πρόταση προάγεται σε υποκείμενο σε μια παθητική πρόταση. Ενώ ο υποβιβασμός είναι όταν ένα στοιχείο μεταβαίνει σε έναν λιγότερο σημαντικό ρόλο. Για παράδειγμα όταν το υποκείμενο μιας ενεργητικής πρότασης που υποβιβάζεται σε προαιρετικό ρόλο σε μια παθητική πρόταση.

## 2.5 Περιορισμοί και Καθολικότητα (Constraints and Universality)

Οι περιορισμοί στη Relational Grammar ρυθμίζουν το πώς οι γραμματικές σχέσεις μπορούν να αλλάξουν μέσα σε μια γλώσσα. Οι περιορισμοί είναι σημαντικοί για τη διατήρηση της γραμματικής συνοχής και διαμορφώνουν τις πιθανές μετατροπές των σχέσεων σε διαφορετικά στρώματα. Στην Καθολικότητα, η Relational Grammar επιχειρεί να προσδιορίσει από τους περιορισμούς ποιοι είναι καθολικοί, δηλαδή σε ποιες γλώσσες εφαρμόζεται και ποιοι σε συγκεκριμένη γλώσσα.

## 2.6 Θεωρητικές Εφαρμογές και Αναλύσεις

Η Relational Grammar χρησιμοποιείται για ανάλυση διαφόρων συντακτικών φαινομένων, όπως:

- **Παθητικοποίηση (Passivization)**: Προάγεται σε υποκείμενο το άμεσο αντικείμενο όταν μετατρέπονται οι ενεργητικές προτάσεις σε παθητικές.
- **Ανύψωση κι Έλεγχος (Raising and Control):** Αναλύουν δομές όπου το υποκείμενο ενός ενθέτου ρήματος "ανυψώνεται" στη θέση υποκειμένου του κύριου ρήματος ή διατηρείται σταθερό σε συντακτικές δομές ελέγχου.
- **Equi:** Αναλύουν δομές που αφορούν την ταυτότητα του υποκειμένου σε διαφορετικά σημεία (κλάσματα) της πρότασης. Συχνά σχετίζονται με ρήματα που απαιτούν απρόσωπα ή ελλειπτικά υποκείμενα.

## 2.7 Κριτική και Εξέλιξη

Χάρις στην πολυπλοκότητα της Relational Grammar και την έλλειψη σαφών μηχανισμών έχει επικριθεί για την αντιμετώπιση όλων των συντακτικών φαινομένων που παρατηρούνται στις γλώσσες. Παρόλα αυτά η Relational Grammar έχει επηρεάσει βαθιά τη σύγχρονη γλωσσολογία. Άνοιξε το δρόμο για άλλες θεωρίες, όπως η Lexical Functional Grammar. Αυτή η θεωρία διατηρεί την εστίαση στις γραμματικές σχέσεις, αλλά ενσωματώνει περισσότερους μηχανισμούς για τη διαχείριση των συντακτικών δομών από ότι η Relational Grammar. Όμως η Relational Grammar παραμένει ένα σημαντικό σημείο αναφοράς, καθώς κατανοεί τους μετασχηματισμούς και τις γραμματικές σχέσεις σε διαφορετικές γλώσσες. Η δράση της συνεχίζει ακόμα να μελετάται ως ένα πλαίσιο που προσφέρει εναλλακτικές οπτικές στη συντακτική ανάλυση.

## 2.8 LoMaFi

```
# Συνάρτηση που αναζηταει καταλήξεις μιας λέξης
def Find_Postfix(lexis):

    size = StringSize(lexis)  # Υπολογισμός μήκους της λέξης
    max_postfix_length = 9  # Μέγιστο μήκος κατάληξης (ιοντουσαν)

    if size <= max_postfix_length:
        range_val = size - 1  # Ρύθμιση εύρους
    else:
        range_val = max_postfix_length

    start = size - range_val + 1  # Υπολογισμός αρχικού σημείου

    all_results = []  # Δημιουργία κενής λίστας για αποθήκευση όλων των αποτελεσμάτ

    for i in range(start, size + 1):  # Επανάληψη για κάθε πιθανή κατάληξη

        lemma = lexis[i - 1:]  # Δημιουργία λήματος
        print("Έλεγχος κατάληξης :", lemma)  # Εκτύπωση λήματος
        found, results = Lexicon(lemma)  # Αναζήτηση στο λεξικό

        if found:
            all_results.extend(results)  # Προσθήκη αποτελεσμάτων στη λίστα
            break  # Διακοπή της επανάληψης αν βρέθηκε αποτέλεσμα, γιατί συνεχίζει

    if all_results:
        return all_results  # Επιστροφή των αποτελεσμάτων αν βρέθηκαν
    else:
        return None  # Επιστροφή μηδενικών αν δεν βρέθηκαν
```

Η παραπάνω συνάρτηση σχεδιάστηκε για να εντοπίζει τις καταλήξεις μιας λέξης. Τα δεδομένα όλων των καταλήξεων βρίσκονται αποθηκευμένα σε ένα αρχείο Excel. Αρχικά υπολογίζει το μήκος μιας λέξης με τη βοήθεια της συνάρτησης StringSize() και αποθηκεύεται σε μια άλλη μεταβλητή για να τη διαχειριστεί αργότερα η συνάρτηση. Στη συνέχεια σε μια άλλη μεταβλητή καθορίζεται το μέγιστο δυνατό μήκος κατάληξης που μπορεί να πάρει μια κατάληξη, το οποίο έχει οριστεί σε 9 χαρακτήρες (για παράδειγμα, η κατάληξη “ιοντουσαν”). Αν το μήκος της εκάστοτε λέξης είναι μικρότερο ή ίσο με το μέγιστο μήκος κατάληξης, τότε το εύρος αναζήτησης ρυθμίζεται να είναι μικρότερο κατά έναν χαρακτήρα από το μήκος της λέξης. Αντιθέτως αν το μήκος της λέξης είναι μεγαλύτερο, το εύρος περιορίζεται στο μέγιστο μήκος κατάληξης.

Εν συνεχεία, υπολογίζει το σημείο εκκίνησης της αναζήτησης για τις καταλήξεις, το οποίο τίθεται από το μέγεθος της λέξης μείον το εύρος αναζήτησης συν ένα και αποθηκεύεται στη μεταβλητή Start. Ακόμη δημιουργείται μια κενή λίστα για την αποθήκευση όλων των καταλήξεων που θα βρεθούν, καθώς ενδέχεται να βρει παραπάνω από μία ίδια κατάληξη. Επίσης η αναζήτηση των καταλήξεων γίνεται μέσω ενός βρόχου. Ο βρόχος αυτός είναι η For, η αναζήτηση αυτή ξεκινά από το Start έως το τέλος της λέξης. Σε κάθε επανάληψη εξετάζει διαφορετικό τμήμα της λέξης, το οποίο αποθηκεύεται ως Lemma. Το λήμμα περιέχει την πιθανή κατάληξη, η οποία στη συνέχεια ελέγχεται αν υπάρχει στο λεξικό με τη βοήθεια της συνάρτησης Lexicon(lemma).

Εφόσον βρεθεί η κατάληξη στο λεξικό, τότε τα αποτελέσματα αποθηκεύονται στην κενή λίστα και η αναζήτηση σταματά, διότι δεν υπάρχει νόημα να συνεχίσει να κόβει χαρακτήρες από τη λέξη. Εάν βρεθούν οι καταλήξεις, η συνάρτηση επιστρέφει τη λίστα

με όλα τα αποτελέσματα. Διαφορετικά επιστρέφει την τιμή None, δηλώνοντας πως δεν βρέθηκε καμία κατάληξη. Η μεθοδική λειτουργία της συνάρτησης εξασφαλίζει την αποτελεσματική αναζήτηση κι εντοπισμό καταλήξεων σε γλωσσικά δεδομένα. Έτσι προσφέρει τη δυνατότητα για περισσότερη ανάλυση κι επεξεργασία στο πλαίσιο μιας εφαρμογής η μιας γλωσσολογικής μελέτης.

# 3. ΣΧΕΔΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

## 3.1 Βάσεις δεδομένων

1. **Microsoft Access**:
   Αυτή η βάση δεδομένων ανήκει στην οικογένεια της Microsoft Office, είναι ιδανική για τη διαχείριση μικρών έως μεσαίων βάσεων δεδομένων.

   - **Υπέρ**: Εύκολο στη χρήση και καλό για αρχάριους.
   - **Κατά**: Δεν είναι κατάλληλο για πολύ μεγάλες βάσεις δεδομένων.

2. **Google Sheets**:

   Είναι ένα εργαλείο της Google που μπορεί να χρησιμοποιηθεί για την αποθήκευση και διαχείριση δεδομένων με παρόμοιο τρόπο όπως το Excel.

   - **Υπέρ**: Εύκολο στη χρήση και μπορεί να χρησιμοποιηθεί για τη δημιουργία και διαχείριση απλών λεξικών δεδομένων.
   - **Κατά**: Έχει περιορισμούς στον όγκο των δεδομένων που μπορεί να χειριστεί.

3. **Excel**:

Είναι ένα λογισμικό υπολογιστικών φύλλων της Microsoft, το οποίο είναι ένα ισχυρό εργαλείο για τη διαχείριση και ανάλυση δεδομένων. Συγκριτικά με τις άλλες βάσεις που προαναφέρθηκαν διαχειρίζεται μεγάλο όγκο δεδομένων, πράγμα που το κάνει ιδανικό για λεξικά δεδομένα. Παρά την αποτελεσματικότητα έχει κάποια υπέρ και κατά:

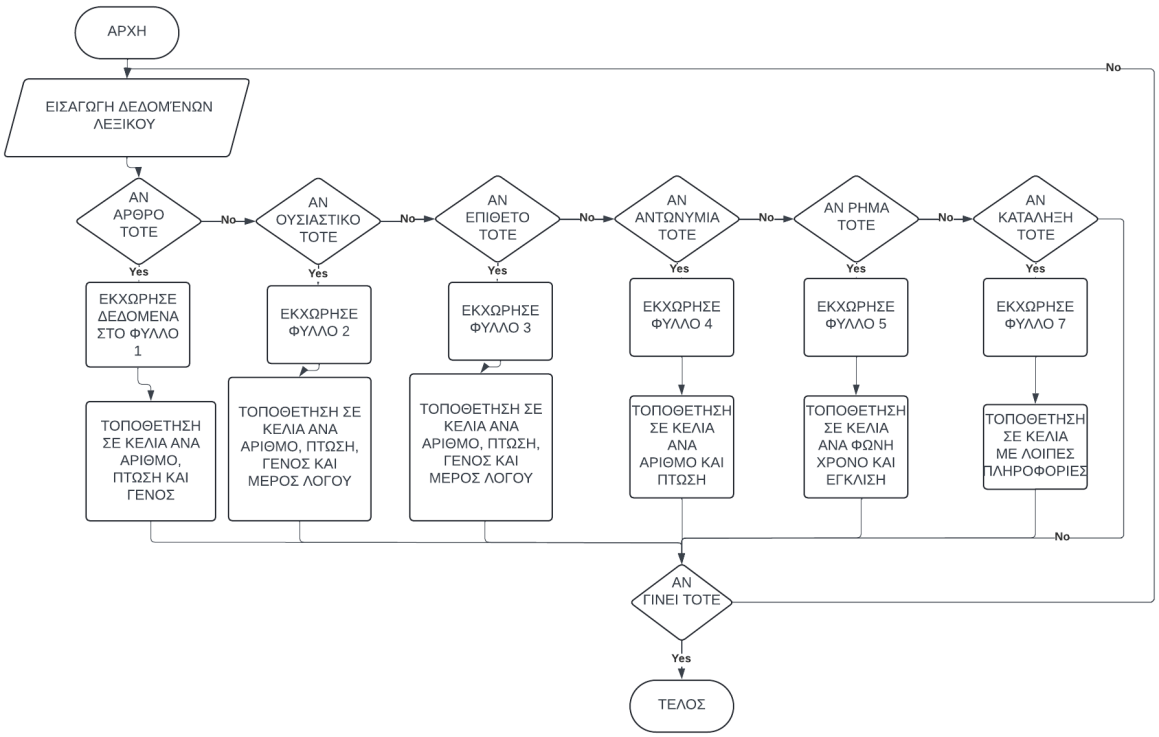
- **Υπέρ**: Εύκολη χρήση που επιτρέπει γρήγορη εκμάθηση και υποστηρίζει ταξινόμηση και συνδυασμό δεδομένων.
- **Κατά**: Προβλήματα απόδοσης σε πολύ μεγάλα σε όγκο δεδομένα, η ακρίβεια των δεδομένων κρίνεται από την προσωπική διαχείριση του χρήστη και οδηγώντας σε πιθανά λάθη.

## 3.2 Επιλογή βάσης δεδομένων

Η επιλογή βάσης δεδομένων ήταν απλή, το Excel είναι ιδανικό για τη διαχείριση ενός λεξικού που περιλαμβάνει άρθρα, αντωνυμίες και άλλους γλωσσικούς πόρους. Το Excel επιτρέπει εύκολη οργάνωση και αναζήτηση δεδομένων. Για παράδειγμα με την πληκτρολόγηση δυο μονάχα κουμπιών βρίσκει αμέσως τη λέξη που χρειαζόμαστε. Παράλληλα διατηρεί μια απλή διεπαφή με τον χρήστη που δεν απαιτεί εξειδικευμένες γνώσεις προγραμματισμού. Η Access χρειάζεται καλές γνώσεις προγραμματισμού, πράγμα που χρειάζεται πολύ χρόνο για την υλοποίηση της βάσης. Επιπλέον το Excel δεν εξαρτάται από τη συνδεσιμότητα και την κοινή χρήση μέσω διαδικτύου, όπως το Google Sheets. Τέλος το λεξικό έχει πολύ μεγάλο όγκο δεδομένων και υποστηρίζεται

πιο εύκολα από το Excel, ενώ οι υπόλοιπες επιλογές διαχειρίζονται μικρότερο όγκο δεδομένων.

## 3.3 Διάγραμμα ροής Excel




## 3.4 Γλώσσες προγραμματισμού

### 1. Python

Μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού είναι η Python. Στον τομέα της τεχνητής νοημοσύνης και της επεξεργασίας φυσικής γλώσσας είναι από τις πιο διαδεδομένες. Η απλότητα και η καθαρή της σύνταξη την καθιστά εύκολη σε πρόσβαση σε προγραμματιστές όλων των επιπέδων. Εκτός από αυτά διαθέτει μια μεγάλη πληθώρα βιβλιοθηκών όπως το TensorFlow, Keras, και PyTorch που επιτρέπει την ανάπτυξη μοντέλων μηχανικής μάθησης. Επιπροσθέτως υπάρχουν βιβλιοθήκες που είναι εξαιρετικά εργαλεία για την ανάλυση και ανάπτυξη γλωσσικών δεδομένων, όπως το NLTK και το spaCy. Επιπλέον η Python παρέχει μεγάλες κοινότητες υποστήριξης, διευκολύνοντας έτσι την ανάπτυξη και τον πειραματισμό.

### 2. C++

Η C++ είναι μια γλώσσα που διαθέτει υψηλή απόδοση και χρησιμοποιείται περισσότερο για την ανάπτυξη εφαρμογών, όπου εκεί η ταχύτητα και η αποδοτικότητα είναι λιγοστές. Ακόμη είναι γνωστή για την ικανότητα να προσφέρει λεπτομερή έλεγχο στη διαχείριση πόρων και μνήμης. Κάτι τέτοιο την καθιστά ιδανική για την τεχνητή νοημοσύνη, που απαιτεί έντονη επεξεργασία δεδομένων. Η C++ χρησιμοποιείται για την ανάπτυξη παιχνιδιών, λειτουργικών συστημάτων κι εφαρμογών. Όλα αυτά απαιτούν υψηλές επιδόσεις όπως και οι αλγόριθμοι μηχανικής μάθησης και

επεξεργασίας εικόνας. Παρόλα τα θετικά της C++ είναι πιο δύσκολη στην εκμάθηση συγκριτικά με την Python και αυτό γιατί είναι πολύπλοκη στη σύνταξη της.

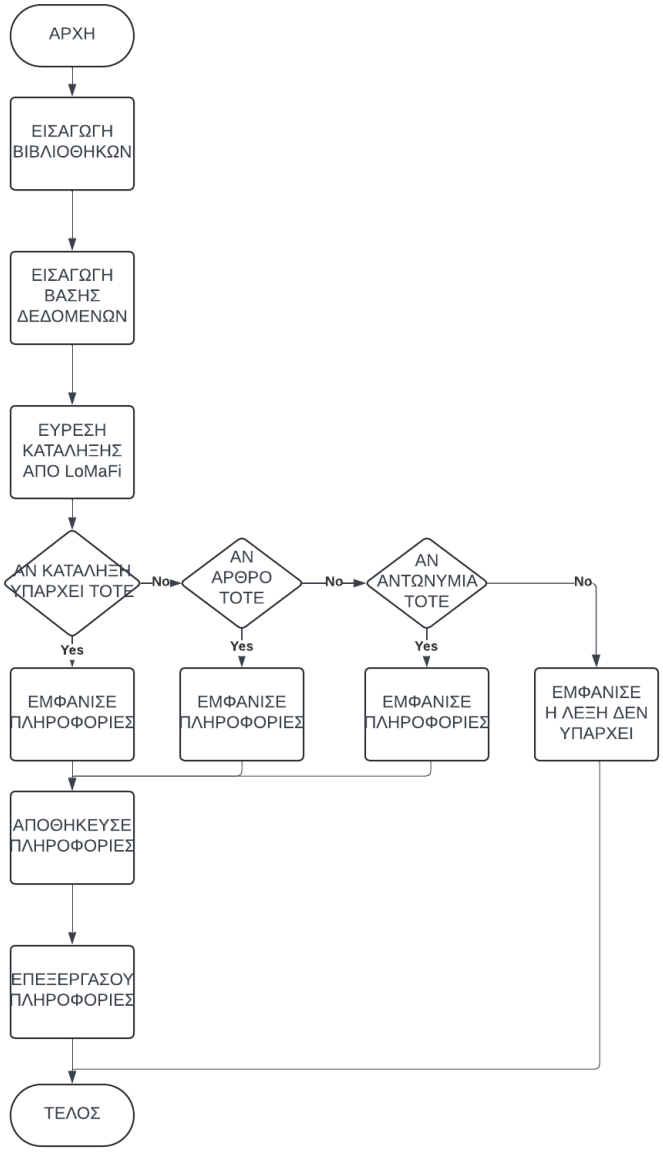
3. **Java**

Η Java επονομάζεται ως μια αντικειμενοστραφής γλώσσα προγραμματισμού, χρησιμοποιείται για την ανάπτυξη επιχειρηματικών εφαρμογών, μεγάλων συστημάτων, και διαδικτυακών εφαρμογών. Επίσης είναι ανεξάρτητη, καθώς οι εφαρμογές της μπορούν να τρέξουν οπουδήποτε σε σύστημα που υποστηρίζει Java Virtual Machine (JVM). Στην επεξεργασία φυσικής γλώσσας και τεχνητής νοημοσύνης προσφέρονται εργαλεία όπως το Deeplearning4j και το Weka. Τα εργαλεία αυτά μπορούν να χρησιμοποιηθούν για την ανάπτυξη και την εκπαίδευση μοντέλων μηχανικής μάθησης. Τέλος η Java είναι δημοφιλής στη δημιουργία συστημάτων που απαιτούν αξιοπιστία κι επεκτασιμότητα, αλλά είναι δύσκολη στη χρήση για γρήγορη ανάπτυξη από την Python.

## 3.5 Επιλογή γλώσσας προγραμματισμού

Η καταλληλότερη γλώσσα προγραμματισμού είναι η Python, παρότι η C++ και η Java προσφέρουν ισχυρά χαρακτηριστικά και απόδοση. Ο λόγος είναι ότι είναι εύκολη στη χρήση και υποστηρίζει πληθώρα βιβλιοθηκών. Επιπλέον θα βοηθήσει στη γρηγορότερη επίλυση του προβλήματος στα καινοτόμα πεδία όπως η τεχνητή νοημοσύνη και η γλωσσολογία. Τα μοντέλα μηχανικής μάθησης διευκολύνουν την εργασία με γλωσσικά δεδομένα και την εφαρμογή σύνθετων αλγορίθμων. Επομένως το πρακτικό κομμάτι θα υλοποιηθεί με τη χρήση της Python καθιστώντας την ιδανική επιλογή για την επίτευξη των στόχων.

## 3.6 Διάγραμμα ροής κώδικα



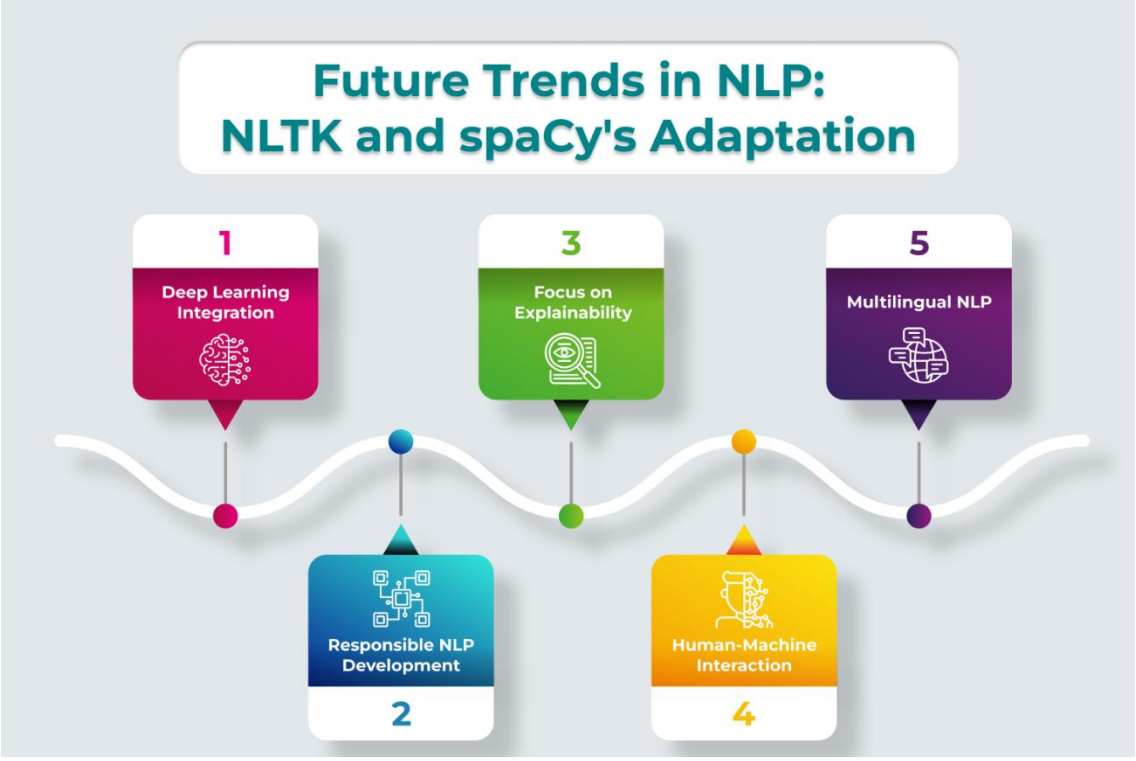

## 3.7 Βιβλιοθήκες

Η **NLTK (Natural Language Toolkit)** είναι μια βιβλιοθήκη στη γλώσσα Python, όπου επεξεργάζεται τη φυσική γλώσσα. Μερικά εργαλεία της περιλαμβάνουν **tokenization** (διαχωρισμός κειμένου σε λέξεις ή προτάσεις), **stemming** (εξαγωγή ριζών λέξεων), **tagging** (αναγνώριση γραμματικών κατηγοριών) και **parsing** (συντακτική ανάλυση). Η βιβλιοθήκη αυτή είναι ιδανική για εκπαιδευτικούς κι ερευνητικούς σκοπούς, επειδή διαθέτει μεγάλη γκάμα μεθόδων και πόρων για τη μελέτη της γλώσσας.

Αντίθετα η **spaCy** είναι μια πιο σύγχρονη βιβλιοθήκη με χαρακτηριστικά υψηλής απόδοσης για την επεξεργασία φυσικής γλώσσας. Σχεδιάστηκε για τη χρήση σε εμπορικές εφαρμογές, ενώ προσφέρει γρήγορα και ακριβή μοντέλα για γλωσσικές εργασίες. Μερικά από αυτά είναι **tokenization, part-of-speech tagging, named entity**

**recognition (NER),** και **dependency parsing.** Επίσης η **spaCy** χρησιμοποιείται σε εφαρμογές που χρειάζεται μεγαλύτερη απόδοση και ακρίβεια.

Μία άλλη βιβλιοθήκη είναι η **Pandas**, είναι ευέλικτη με βαθιά ανάλυση που διαχειρίζεται δεδομένα. Οι δομές δεδομένων που παρέχει είναι ισχυρές, όπως τα DataFrames. Τα DataFrames επιτρέπουν εύκολη αποθήκευση, επεξεργασία και ανάλυση δεδομένων. Παρότι δεν είναι εξειδικευμένη όπως οι άλλες δυο βιβλιοθήκες, χρησιμοποιείται σε συνδυασμό με βιβλιοθήκες Natural language processing. Επομένως την καθιστά ένα σημαντικό εργαλείο στη γλωσσολογία.




(Πηγή: https://www.seaflux.tech/blogs/nltk-vs-spacy-nlp-libraries-comparison)

# 4. ΥΛΟΠΟΙΗΣΗ - ΚΑΤΑΣΚΕΥΕΣ

## 4.1 Excel

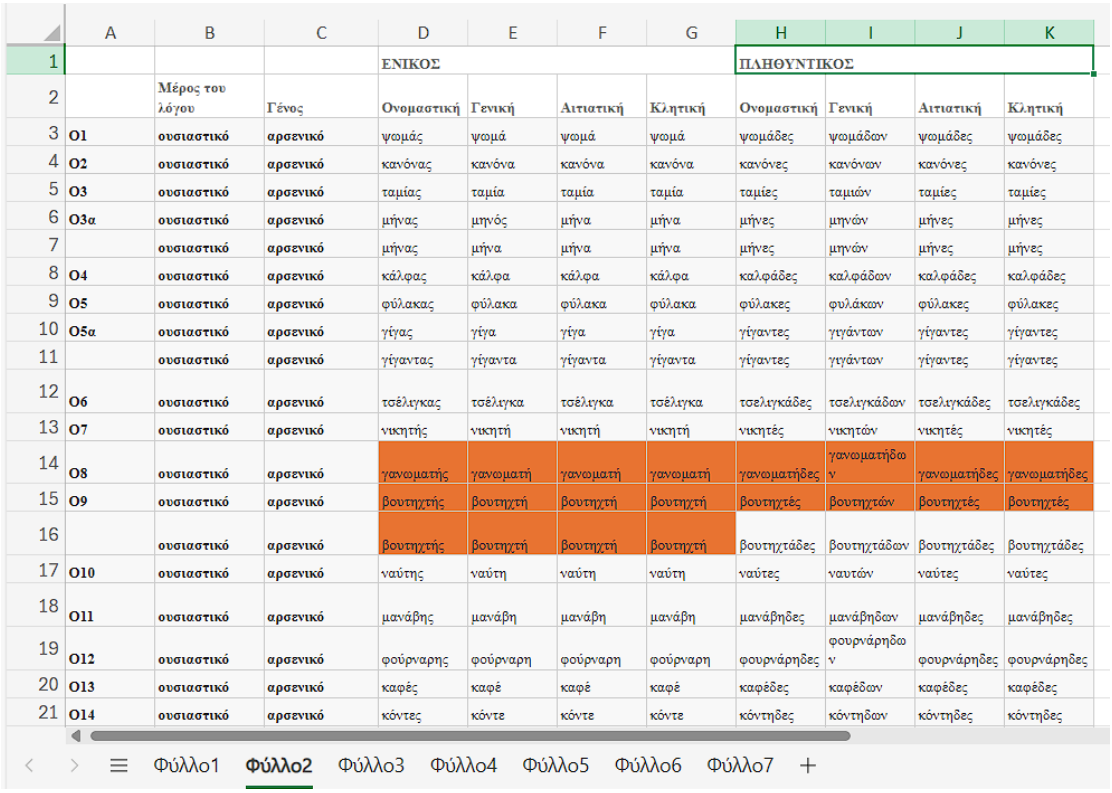
Κατά την υλοποίηση της βάσης δεδομένων δεν προστέθηκαν όπως είναι τα δεδομένα από το λεξικό του Τριανταφυλλίδη. Στη βάση προστέθηκαν ανά κελί πληροφορίες που θα διευκόλυνε τον κώδικα να εμφανίσει στοιχεία στο κύριο πρόγραμμα.

**Άρθρο**

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | ΕΝΙΚΟΣ | ΕΝΙΚΟΣ | ΕΝΙΚΟΣ | ΕΝΙΚΟΣ | ΠΛΗΘΥΝΤΙΚΟΣ | ΠΛΗΘΥΝΤΙΚΟΣ | ΠΛΗΘΥΝΤΙΚΟΣ | ΠΛΗΘΥΝΤΙΚΟΣ |
| 2 | | Ονομαστική | Γενική | Αιτιατική | Κλητική | Ονομαστική | Γενική | Αιτιατική | Κλητική |
| 3 | **Αρσ.** | ο | του | το(ν) | # | οι | των | τους | # |
| 4 | **Αρσ.** | ο | στου | στον | # | οι | στων | στους | # |
| 5 | **Θηλ.** | η | της | τη | # | οι | των | τις | # |
| 6 | **Θηλ.** | η | της | την | # | οι | των | τις | # |
| 7 | **Θηλ.** | η | στης | στην | # | οι | στων | στις | # |
| 8 | **Ουδ.** | το | του | το | # | τα | των | τα | |

Στην παραπάνω εικόνα έχουν προστεθεί όλα τα άρθρα από το λεξικό. Μερικά άρθρα είναι δύο φορές γιατί στην γενική πτώση υπάρχουν διαφορετικές παραλλαγές. Έτσι όταν για παράδειγμα αναζητάει το άρθρο “στου”, να εμφανίζει όλα τα στοιχεία της γραμμής και στήλης. Αν το “στου” έμπαινε με “/” θα εμφάνιζε και τα δύο μαζί ή δεν θα έβρισκε το “στου”. Επιπλέον όπου δεν υπάρχει άρθρο έχει προστεθεί η δίεση για να εκτυπώνει μήνυμα ότι δεν υπάρχει τέτοιο άρθρο. Στην συνέχεια στην πρώτη γραμμή έχει γραφτεί ο αριθμός σε κάθε κελί, διότι αν έμπαινε μόνο στην ονομαστική θα εμφάνιζε κενό. Επίσης στην δεύτερη γραμμή έχει προστεθεί η πτώση του κάθε άρθρου, η οποία θα χρησιμοποιείται στο πρόγραμμα όπως και τα υπόλοιπα. Τέλος στην πρώτη στήλη υπάρχει και το γένος του κάθε άρθρου.

## Ουσιαστικά

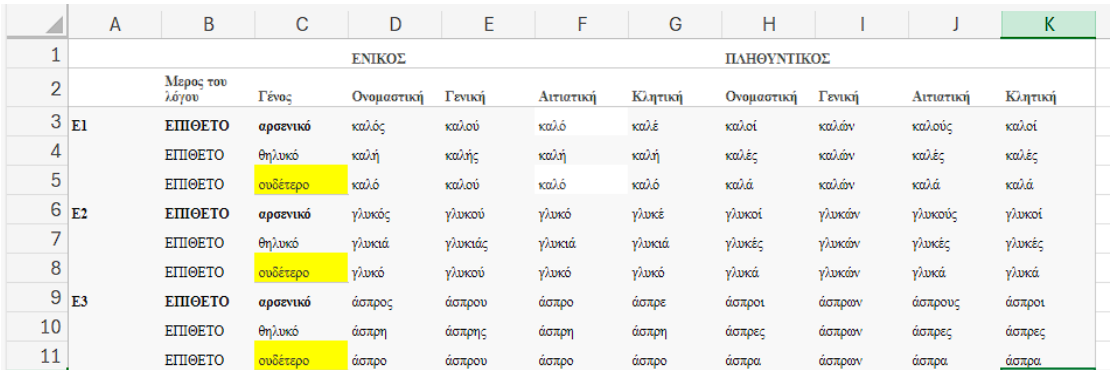


| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | ΕΝΙΚΟΣ | | | | ΠΛΗΘΥΝΤΙΚΟΣ | | | |
| 2 | | Μέρος του λόγου | Γένος | Ονομαστική | Γενική | Αιτιατική | Κλητική | Ονομαστική | Γενική | Αιτιατική | Κλητική |
| 3 | **Ο1** | **ουσιαστικό** | **αρσενικό** | ψωμάς | ψωμά | ψωμά | ψωμά | ψωμάδες | ψωμάδων | ψωμάδες | ψωμάδες |
| 4 | **Ο2** | **ουσιαστικό** | **αρσενικό** | κανόνας | κανόνα | κανόνα | κανόνα | κανόνες | κανόνων | κανόνες | κανόνες |
| 5 | **Ο3** | **ουσιαστικό** | **αρσενικό** | ταμίας | ταμία | ταμία | ταμία | ταμίες | ταμιών | ταμίες | ταμίες |
| 6 | **Ο3α** | **ουσιαστικό** | **αρσενικό** | μήνας | μηνός | μήνα | μήνα | μήνες | μηνών | μήνες | μήνες |
| 7 | | **ουσιαστικό** | **αρσενικό** | μήνας | μήνα | μήνα | μήνα | μήνες | μηνών | μήνες | μήνες |
| 8 | **Ο4** | **ουσιαστικό** | **αρσενικό** | κάλφας | κάλφα | κάλφα | κάλφα | καλφάδες | καλφάδων | καλφάδες | καλφάδες |
| 9 | **Ο5** | **ουσιαστικό** | **αρσενικό** | φύλακας | φύλακα | φύλακα | φύλακα | φύλακες | φυλάκων | φύλακες | φύλακες |
| 10 | **Ο5α** | **ουσιαστικό** | **αρσενικό** | γίγας | γίγα | γίγα | γίγα | γίγαντες | γιγάντων | γίγαντες | γίγαντες |
| 11 | | **ουσιαστικό** | **αρσενικό** | γίγαντας | γίγαντα | γίγαντα | γίγαντα | γίγαντες | γιγάντων | γίγαντες | γίγαντες |
| 12 | **Ο6** | **ουσιαστικό** | **αρσενικό** | τσέλιγκας | τσέλιγκα | τσέλιγκα | τσέλιγκα | τσελιγκάδες | τσελιγκάδων | τσελιγκάδες | τσελιγκάδες |
| 13 | **Ο7** | **ουσιαστικό** | **αρσενικό** | νικητής | νικητή | νικητή | νικητή | νικητές | νικητών | νικητές | νικητές |
| 14 | **Ο8** | **ουσιαστικό** | **αρσενικό** | γανωματής | γανωματή | γανωματή | γανωματή | γανωματήδες | γανωματήδων | γανωματήδες | γανωματήδες |
| 15 | **Ο9** | **ουσιαστικό** | **αρσενικό** | βουτηχτής | βουτηχτή | βουτηχτή | βουτηχτή | βουτηχτές | βουτηχτών | βουτηχτές | βουτηχτές |
| 16 | | **ουσιαστικό** | **αρσενικό** | βουτηχτής | βουτηχτή | βουτηχτή | βουτηχτή | βουτηχτάδες | βουτηχτάδων | βουτηχτάδες | βουτηχτάδες |
| 17 | **Ο10** | **ουσιαστικό** | **αρσενικό** | ναύτης | ναύτη | ναύτη | ναύτη | ναύτες | ναυτών | ναύτες | ναύτες |
| 18 | **Ο11** | **ουσιαστικό** | **αρσενικό** | μανάβης | μανάβη | μανάβη | μανάβη | μανάβηδες | μανάβηδων | μανάβηδες | μανάβηδες |
| 19 | **Ο12** | **ουσιαστικό** | **αρσενικό** | φούρναρης | φούρναρη | φούρναρη | φούρναρη | φουρνάρηδες | φουρνάρηδων | φουρνάρηδες | φουρνάρηδες |
| 20 | **Ο13** | **ουσιαστικό** | **αρσενικό** | καφές | καφέ | καφέ | καφέ | καφέδες | καφέδων | καφέδες | καφέδες |
| 21 | **Ο14** | **ουσιαστικό** | **αρσενικό** | κόντες | κόντε | κόντε | κόντε | κόντηδες | κόντηδων | κόντηδες | κόντηδες |

Φύλλο1 **Φύλλο2** Φύλλο3 Φύλλο4 Φύλλο5 Φύλλο6 Φύλλο7

Στα ουσιαστικά υπάρχουν σχεδόν οι ίδιες προσθήκες όπως στα άρθρα. Πρώτα απ' όλα έχει προστεθεί το μέρος του λόγου στη δεύτερη στήλη και αυτό γιατί μερικές καταλήξεις σε "άς" είναι κοινά σε ουσιαστικά κι επίθετα. Έτσι αν βρίσκει την ίδια κατάληξη, να εμφανίζει το σωστό μέρος του λόγου. Ακόμη στην τρίτη στήλη έχει προστεθεί το γένος και πτώση της κάθε λέξης, όπως στα άρθρα. Επίσης όπου δεν υπάρχει ουσιαστικό έχει προστεθεί η δίεση, ενώ ο αριθμός δεν έχει αλλάξει επειδή ο αριθμός φαίνεται από τις καταλήξεις. Τέλος τα χρωματισμένα πεδία δεν καθορίζουν τίποτα, απλά έχουν προστεθεί για πρακτικούς λόγους.

## Επίθετα και μετοχές



| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | ΕΝΙΚΟΣ | | | | ΠΛΗΘΥΝΤΙΚΟΣ | | | |
| 2 | | Μέρος του λόγου | Γένος | Ονομαστική | Γενική | Αιτιατική | Κλητική | Ονομαστική | Γενική | Αιτιατική | Κλητική |
| 3 | **Ε1** | **ΕΠΙΘΕΤΟ** | **αρσενικό** | καλός | καλού | καλό | καλέ | καλοί | καλών | καλούς | καλοί |
| 4 | | ΕΠΙΘΕΤΟ | θηλυκό | καλή | καλής | καλή | καλή | καλές | καλών | καλές | καλές |
| 5 | | ΕΠΙΘΕΤΟ | ουδέτερο | καλό | καλού | καλό | καλό | καλά | καλών | καλά | καλά |
| 6 | **Ε2** | **ΕΠΙΘΕΤΟ** | **αρσενικό** | γλυκός | γλυκού | γλυκό | γλυκέ | γλυκοί | γλυκών | γλυκούς | γλυκοί |
| 7 | | ΕΠΙΘΕΤΟ | θηλυκό | γλυκιά | γλυκιάς | γλυκιά | γλυκιά | γλυκές | γλυκών | γλυκές | γλυκές |
| 8 | | ΕΠΙΘΕΤΟ | ουδέτερο | γλυκό | γλυκού | γλυκό | γλυκό | γλυκά | γλυκών | γλυκά | γλυκά |
| 9 | **Ε3** | **ΕΠΙΘΕΤΟ** | **αρσενικό** | άσπρος | άσπρου | άσπρο | άσπρε | άσπροι | άσπρων | άσπρους | άσπροι |
| 10 | | ΕΠΙΘΕΤΟ | θηλυκό | άσπρη | άσπρης | άσπρη | άσπρη | άσπρες | άσπρων | άσπρες | άσπρες |
| 11 | | ΕΠΙΘΕΤΟ | ουδέτερο | άσπρο | άσπρου | άσπρο | άσπρο | άσπρα | άσπρων | άσπρα | άσπρα |

Στα επίθετα και μετοχές δεν υπάρχει καμία παραπάνω αλλαγή από τα ουσιαστικά, είναι ακριβώς ίδια.

### Αντωνυμίες

| | ΕΝΙΚΟΣ | ΕΝΙΚΟΣ | ΕΝΙΚΟΣ | ΕΝΙΚΟΣ | ΠΛΗΘΥΝΤΙΚΟΣ | ΠΛΗΘΥΝΤΙΚΟΣ | ΠΛΗΘΥΝΤΙΚΟΣ | ΠΛΗΘΥΝΤΙΚΟΣ |
|---|---|---|---|---|---|---|---|---|
| 3 | Ονομαστική | Γενική | Αιτιατική | Κλητική | Ονομαστική | Γενική | Αιτιατική | Κλητική |
| 4 | εγώ | εμένα | εμένα | # | εμείς | εμάς | εμάς | # |
| 5 | # | μου | με | # | | μας | μας | # |
| 6 | | | | | | | | |
| 7 | εσύ | εσένα | εσένα | # | εσείς | εσάς | εσάς | # |
| 8 | # | σου | σε | # | | σας | σας | # |
| 9 | | | | | | | | |
| 10 | αυτός | αυτού | αυτόν | # | αυτοί | αυτών | αυτούς | # |
| 11 | τος | του | τον | # | τοι | τους | τους | # |
| 12 | αυτή | αυτής | αυτή(ν) | # | αυτές | αυτών | αυτές | # |
| 13 | τη | της | τη(ν) | # | τες | τους | τις, τες | # |
| 14 | αυτό | αυτού | αυτό | # | αυτά | αυτών | αυτά | # |
| 15 | το | του | το | # | τα | τους | τα | # |

Στις αντωνυμίες υπάρχει η ίδια ακριβώς μορφή με τα άρθρα, πρώτη γραμμή αριθμός, δεύτερη γραμμή πτώση και όπου δεν υπάρχει αντωνυμία βρίσκεται στην θέση της η δίεση.

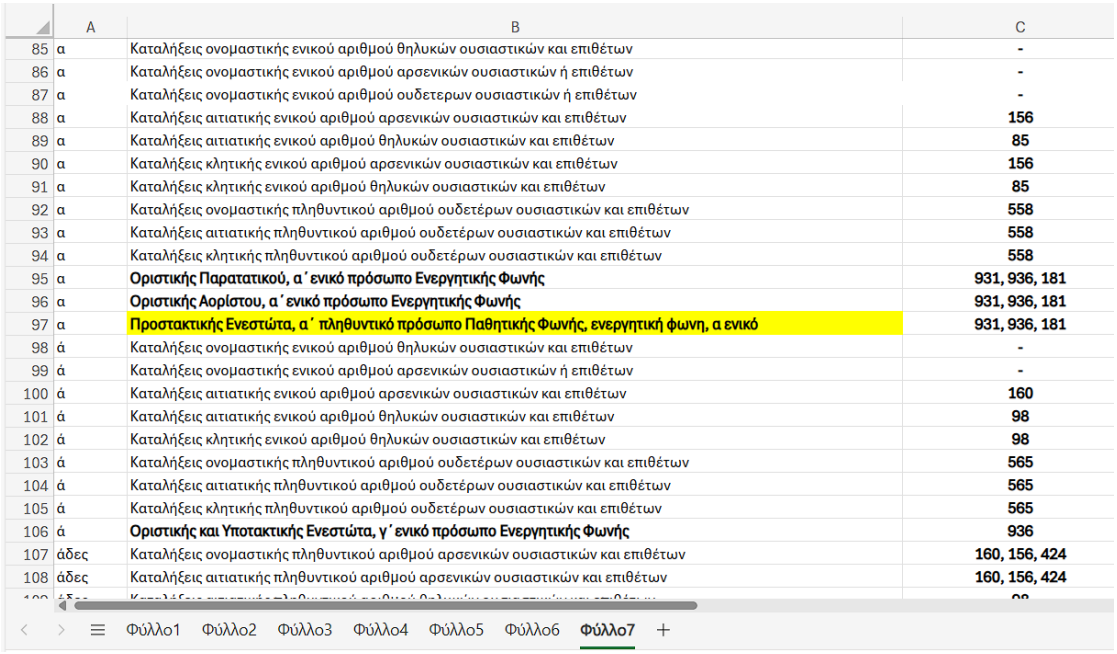
### Ρήματα

| | | P1α Ενεργητική φωνή | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Ενεργητική | ενεστώτας | οριστ. /υποτ. | | | κλειδώνω | κλειδώνεις | κλειδώνει | κλειδώνο(υ)με | κλειδώνετε | κλειδώνουν |
| 3 | Ενεργητική | ενεστώτας | προστ. | | | | κλειδωνε | | | κλειδώνετε | |
| 4 | Ενεργητική | ενεστώτας | μτχ. | | | κλειδώνοντας | | | | | |
| 5 | Ενεργητική | πρτ. | οριστ. | | | κλειδωνα | κλείδωνες | κλείδωνε | κλειδώναμε | κλειδώνατε | κλείδωναν |
| 6 | Ενεργητική | αόρ. | οριστ. | | | κλειδωσα | κλειδώσες | κλειδωσε | κλειδώσαμε | κλειδώσατε | κλείδωσαν |
| 7 | Ενεργητική | αόρ. | υποτ. | | | κλειδώσω | κλειδώσεις | κλειδώσει | κλειδώσο(υ)με | κλειδώσετε | κλειδώσουν |
| 8 | Ενεργητική | αόρ. | προστ. | | | κλείδωσε | κλειδώστε | | | | |
| 9 | Ενεργητική | αόρ. | απαρέμφ. | | | κλειδώσει | | | | | |
| 10 | Ενεργητική | πρκ. | οριστ. | | | έχω κλειδώσει (ή έχω κλειδωμένο) | | | | | |
| 11 | Ενεργητική | πρκ. | υποτ. | | | να έχω κλειδώσει (ή να έχω κλειδωμένο) | | | | | |
| 12 | Ενεργητική | εξακολ. μέλλ. | οριστ. | | | θα κλειδώνω | | | | | |
| 13 | Ενεργητική | στιγμ. μέλλ. | οριστ. | | | θα κλειδώσω | | | | | |
| 14 | Ενεργητική | υπερσ. | οριστ. | | | είχα κλειδώσει ( ή είχα κλειδωμένο ) | | | | | |
| 15 | Ενεργητική | συντελ. μέλλ. | οριστ. | | | θα έχω κλειδώσει (ή θα έχω κλειδωμένο) | | | | | |

Στα ρήματα οι μόνες αλλαγές που έχουν σημειωθεί είναι στην πρώτη, δεύτερη και τρίτη στήλη. Στην πρώτη στήλη βρίσκεται η φωνή, καθώς μερικές καταλήξεις βρίσκονται και στην ενεργητική και στην παθητική (“ει” βρίσκεται στην ενεργητική φωνή του χρόνου ενεστώτα και στην παθητική φωνή του χρόνου αορίστου με έγκλιση απαρεμφάτου). Επιπροσθέτως στη δεύτερη στήλη βρίσκεται ο χρόνος και στην τρίτη στήλη η έγκλιση. Στη δεύτερη και τρίτη στήλη τα στοιχεία θα χρησιμεύσουν ως πληροφορίες, που θα εμφανίζονται στο κύριο πρόγραμμα.

Στο φύλλο 6 βρίσκονται όλες οι καταλήξεις συγκεντρωμένες, χωρίς καμία διάταξη. Στο επόμενο φύλλο βρίσκονται όλες οι καταλήξεις σε αλφαβητική σειρά, όπου θα αναλυθούν παρακάτω.

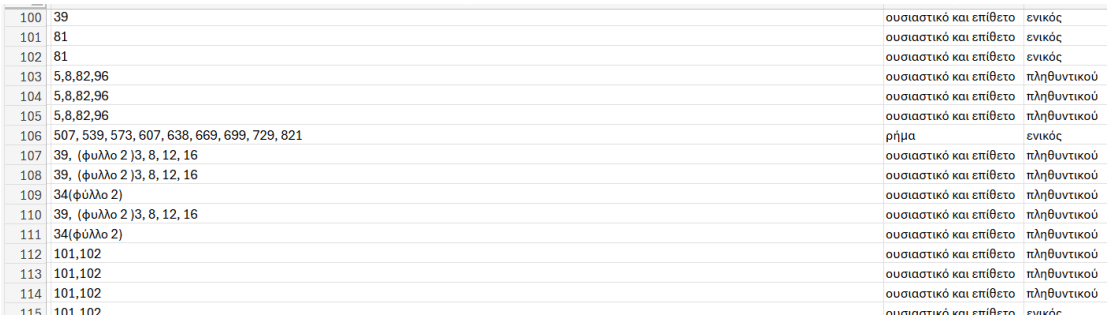
## Καταλήξεις



| | A | B | C |
|---|---|---|---|
| 85 | α | Καταλήξεις ονομαστικής ενικού αριθμού θηλυκών ουσιαστικών και επιθέτων | - |
| 86 | α | Καταλήξεις ονομαστικής ενικού αριθμού αρσενικών ουσιαστικών ή επιθέτων | - |
| 87 | α | Καταλήξεις ονομαστικής ενικού αριθμού ουδετερων ουσιαστικών ή επιθέτων | - |
| 88 | α | Καταλήξεις αιτιατικής ενικού αριθμού αρσενικών ουσιαστικών και επιθέτων | 156 |
| 89 | α | Καταλήξεις αιτιατικής ενικού αριθμού θηλυκών ουσιαστικών και επιθέτων | 85 |
| 90 | α | Καταλήξεις κλητικής ενικού αριθμού αρσενικών ουσιαστικών και επιθέτων | 156 |
| 91 | α | Καταλήξεις κλητικής ενικού αριθμού θηλυκών ουσιαστικών και επιθέτων | 85 |
| 92 | α | Καταλήξεις ονομαστικής πληθυντικού αριθμού ουδετέρων ουσιαστικών και επιθέτων | 558 |
| 93 | α | Καταλήξεις αιτιατικής πληθυντικού αριθμού ουδετέρων ουσιαστικών και επιθέτων | 558 |
| 94 | α | Καταλήξεις κλητικής πληθυντικού αριθμού ουδετέρων ουσιαστικών και επιθέτων | 558 |
| 95 | α | **Οριστικής Παρατατικού, α΄ ενικό πρόσωπο Ενεργητικής Φωνής** | 931, 936, 181 |
| 96 | α | **Οριστικής Αορίστου, α΄ ενικό πρόσωπο Ενεργητικής Φωνής** | 931, 936, 181 |
| 97 | α | **Προστακτικής Ενεστώτα, α΄ πληθυντικό πρόσωπο Παθητικής Φωνής, ενεργητική φωνη, α ενικό** | 931, 936, 181 |
| 98 | ά | Καταλήξεις ονομαστικής ενικού αριθμού θηλυκών ουσιαστικών και επιθέτων | - |
| 99 | ά | Καταλήξεις ονομαστικής ενικού αριθμού αρσενικών ουσιαστικών ή επιθέτων | - |
| 100 | ά | Καταλήξεις αιτιατικής ενικού αριθμού αρσενικών ουσιαστικών και επιθέτων | 160 |
| 101 | ά | Καταλήξεις αιτιατικής ενικού αριθμού θηλυκών ουσιαστικών και επιθέτων | 98 |
| 102 | ά | Καταλήξεις κλητικής ενικού αριθμού θηλυκών ουσιαστικών και επιθέτων | 98 |
| 103 | ά | Καταλήξεις ονομαστικής πληθυντικού αριθμού ουδετέρων ουσιαστικών και επιθέτων | 565 |
| 104 | ά | Καταλήξεις αιτιατικής πληθυντικού αριθμού ουδετέρων ουσιαστικών και επιθέτων | 565 |
| 105 | ά | Καταλήξεις κλητικής πληθυντικού αριθμού ουδετέρων ουσιαστικών και επιθέτων | 565 |
| 106 | ά | **Οριστικής και Υποτακτικής Ενεστώτα, γ΄ ενικό πρόσωπο Ενεργητικής Φωνής** | 936 |
| 107 | άδες | Καταλήξεις ονομαστικής πληθυντικού αριθμού αρσενικών ουσιαστικών και επιθέτων | 160, 156, 424 |
| 108 | άδες | Καταλήξεις αιτιατικής πληθυντικού αριθμού αρσενικών ουσιαστικών και επιθέτων | 160, 156, 424 |

Φύλλο1 Φύλλο2 Φύλλο3 Φύλλο4 Φύλλο5 Φύλλο6 **Φύλλο7**

Στην πρώτη στήλη βρίσκονται όλες οι καταλήξεις συγκεντρωτικά και με αλφαβητική σειρά. Στη συνέχεια στη δεύτερη στήλη αναγράφονται οι πληροφορίες της κάθε κατάληξης, όπως πτώση, αριθμός, γένους και μέρος του λόγου. Στα ρήματα υπάρχει έντονη γραφή για να διακρίνεται από τα υπόλοιπα (ουσιαστικά κι επίθετα) και οι πληροφορίες που παρέχει είναι η έγκλιση, χρόνος, αριθμός, πρόσωπο και φωνή. Ακόμη στην τρίτη στήλη βρίσκεται ο τύπος αναφοράς, ο τύπος αναφοράς είναι η ονομαστική ενικού στα επίθετα και ουσιαστικά, ενώ στα ρήματα το α΄ ενικό πρόσωπο του ενικού αριθμού. Εκεί που υπάρχει ο χαρακτήρας "-" είναι η ίδια γραμμή ο τύπος αναφοράς, εκεί που υπάρχει αριθμός είναι στη συγκεκριμένη σειρά ο τύπος αναφοράς. Για παράδειγμα αν στο κελί του τύπου αναφοράς βρίσκεται ο αριθμός 160, τότε η σειρά 160 είναι η ονομαστική ενικού για τα ουσιαστικά κι επίθετα. Επίσης ο τύπος αναφοράς δεν είναι μοναδικός και μπορεί να είναι παραπάνω από μία φορές στο Excel.



| | | | |
|---|---|---|---|
| 100 | 39 | ουσιαστικό και επίθετο | ενικός |
| 101 | 81 | ουσιαστικό και επίθετο | ενικός |
| 102 | 81 | ουσιαστικό και επίθετο | ενικός |
| 103 | 5,8,82,96 | ουσιαστικό και επίθετο | πληθυντικού |
| 104 | 5,8,82,96 | ουσιαστικό και επίθετο | πληθυντικού |
| 105 | 5,8,82,96 | ουσιαστικό και επίθετο | πληθυντικού |
| 106 | 507, 539, 573, 607, 638, 669, 699, 729, 821 | ρήμα | ενικός |
| 107 | 39, (φυλλο 2 )3, 8, 12, 16 | ουσιαστικό και επίθετο | πληθυντικού |
| 108 | 39, (φυλλο 2 )3, 8, 12, 16 | ουσιαστικό και επίθετο | πληθυντικού |
| 109 | 34(φύλλο 2) | ουσιαστικό και επίθετο | πληθυντικού |
| 110 | 39, (φυλλο 2 )3, 8, 12, 16 | ουσιαστικό και επίθετο | πληθυντικού |
| 111 | 34(φύλλο 2) | ουσιαστικό και επίθετο | πληθυντικού |
| 112 | 101,102 | ουσιαστικό και επίθετο | πληθυντικού |
| 113 | 101,102 | ουσιαστικό και επίθετο | πληθυντικού |
| 114 | 101,102 | ουσιαστικό και επίθετο | πληθυντικού |

Στην τέταρτη στήλη βρίσκονται τα κλητικά παραδείγματα, δηλαδή πού υπάρχει αυτή η κατάληξη στα προηγούμενα φύλλα. Για παράδειγμα η κατάληξη της σειράς 112 βρίσκεται στην σειρά 101 και 102 του φύλλου 3. Τα κλητικά παραδείγματα μπορεί να είναι μοναδικά αλλά και πάρα πολλά. Ακόμη να προστεθεί ότι στα ρήματα τα κλητικά παραδείγματα είναι μόνο στο πέμπτο φύλλο και στα ουσιαστικά κι επίθετα βρίσκονται στο 2ο & 3ο φύλλο.

Στη συνέχεια στην πέμπτη στήλη βρίσκεται το μέρος του λόγου και δίπλα στην 6η στήλη ο αριθμός.

| | | | | | | |
|---|---|---|---|---|---|---|
| 101 | # | # | # | # | αιτιατικής | θηλυκό |
| 102 | # | # | # | # | κλητικής | θηλυκό |
| 103 | # | # | # | # | ονομαστική | ουδέτερο |
| 104 | # | # | # | # | αιτιατικής | ουδέτερο |
| 105 | # | # | # | # | κλητικής | ουδέτερο |
| 106 | ενεργητική | οριστική και υποτακτικής | ενεστώτας | γ | # | # |
| 107 | # | # | # | # | ονομαστική | αρσενικό |
| 108 | # | # | # | # | αιτιατικής | αρσενικό |
| 109 | # | # | # | # | αιτιατικής | θηλυκό |
| 110 | # | # | # | # | κλητικής | αρσενικό |
| 111 | # | # | # | # | κλητικής | αρσενικό |
| 112 | # | # | # | # | ονομαστική | ουδέτερο |
| 113 | # | # | # | # | αιτιατικής | ουδέτερο |
| 114 | # | # | # | # | κλητικής | ουδέτερο |
| 115 | # | # | # | # | ονομαστική | ουδέτερο |
| 116 | # | # | # | # | αιτιατικής | ουδέτερο |
| 117 | # | # | # | # | κλητικής | ουδέτερο |
| 118 | # | # | # | # | γενικής | ουδέτερο |
| 119 | # | # | # | # | γενικής | αρσενικό |
| 120 | # | # | # | # | γενικής | θηλυκό |
| 121 | ενεργητική | οριστική και υποτακτικής | ενεστώτας | γ | # | # |
| 122 | # | # | # | # | ονομαστική | ουδέτερο |

Τέλος η 7η στήλη καθορίζει τη φωνή, η 8η την έγκλιση, η 9η τον χρόνο και η 10η το πρόσωπο. Φυσικά εκεί όπου υπάρχουν οι διέσεις είναι ουσιαστικά ή επίθετα. Επίσης η 11η στήλη είναι η πτώση και στην 12η στήλη το γένος. Εκεί που υπάρχουν διέσεις είναι ρήματα, αφού δεν έχουν γένος και πτώση.

Συνεπώς όλες αυτές οι πληροφορίες μετά την τέταρτη στήλη είναι ξεχωριστές ανά κατάληξη για να προσφέρουν είτε πληροφορία στον κώδικα, είτε επεξεργασία. Στην επεξεργασία άλλοτε γίνεται σύγκριση και άλλοτε επαλήθευση.

## 4.2 Κώδικας

Στον κώδικα υπάρχουν πολλές συναρτήσεις, οι οποίες κατασκευάστηκαν για ευκολότερη διαχείριση και μείωση λαθών. Σχεδόν όλες οι συναρτήσεις είναι ανεξάρτητες μεταξύ τους, αλλά διαχειρίζονται τα ίδια δεδομένα. Η μεθοδολογία αυτή εξασφάλισε χρόνο με αποτέλεσμα το πρακτικό κομμάτι να υλοποιηθεί γρηγορότερα.

### Βιβλιοθήκες

Μια βιβλιοθήκη είναι ένα σύνολο έτοιμων λειτουργιών, οι οποίες έχουν κατασκευαστεί από έναν άλλο προγραμματιστή προς διευκόλυνση κάποιου άλλου. Αυτή η διευκόλυνση μπορεί να αναπτύξει το λογισμικό του εκάστοτε προγραμματιστή απλά με ένα κάλεσμά της. Οι βιβλιοθήκες εκτός από διευκόλυνση προσφέρουν και εξοικονόμηση χώρου. Για παράδειγμα μια βιβλιοθήκη μπορεί να έχει χίλιες γραμμές και με ένα κάλεσμα της μιας γραμμής λύνει τα χέρια του προγραμματιστή.

Για να καλέσεις μια βιβλιοθήκη πρέπει πρώτα να την εισάγεις στο πρόγραμμα. Με τη βοήθεια της εντολής import εισάγει τη βιβλιοθήκη στο πρόγραμμα και μπορούν να χρησιμοποιηθούν οι λειτουργίες της. Οι βιβλιοθήκες που χρησιμοποιήθηκαν στο πρόγραμμα είναι:

- **Pandas:** είναι μια ισχυρή βιβλιοθήκη ιδανική για διαχείριση δεδομένων και χειρισμό πινάκων δεδομένων (DataFrames). Συχνά χρησιμοποιείται για ανάγνωση κι επεξεργασία δεδομένων από αρχεία Excel. Η Pandas μπορεί να επεξεργαστεί στοιχεία σε κάθε γραμμή ή στήλη ή ακόμα και τα δύο μαζί στο Excel, με τη βοήθεια κάποιων εντολών.
- **Re:** επιτρέπει την αναζήτηση και τον χειρισμό συμβολοσειρών με πολύπλοκα μοτίβα. Επιπλέον μερικές από τις λειτουργίες της μπορεί να είναι ο διαχωρισμός αριθμών από συμβολοσειρά, με βάση τα κόμματα ή τα κενά διαστήματα.
- **Unicodedata:** παρέχει εργαλεία για επεξεργασία unicode χαρακτήρων. Μερικές από τις λειτουργίες της είναι αφαίρεση τόνων, η μετατροπή σε μικρά από κεφαλαία γράμματα και άλλα πολλά με τη χρήση φυσικά εντολών.

### Συναρτήσεις

Μια συνάρτηση είναι ένα κομμάτι κώδικα που εκτελεί μια συγκεκριμένη λειτουργία, μια η και περισσότερες φορές σε ένα πρόγραμμα. Οι συναρτήσεις καθιστούν πιο εύκολο, καθαρό και δομημένο στην συντήρηση κώδικα. Για να καλέσεις μία συνάρτηση χρειάζεται να έχει και κάποιο όνομα. Όταν την καλέσεις με το όνομα, θα υπάρχουν και κάποιοι παράμετροι. Οι παράμετροι είναι δεδομένα που μπορείς να περάσεις μέσα στην συνάρτηση, έτσι ώστε να τα επεξεργάζεται. Στο εσωτερικό του έχει κώδικα που διαχειρίζεται τα δεδομένα και παράγει ένα αποτέλεσμα. Το αποτέλεσμα αυτό χρησιμοποιείται στο κύριο πρόγραμμα και εξέρχεται από την συνάρτηση με την εντολή return, αν ο προγραμματιστής το επιθυμεί.

```
11
12  # Συνάρτηση που υπολογίζει το μήκος μιας λέξης
13  def StringSize(lexis):
14      return len(lexis)
15
16  # Συνάρτηση που αναζηταει ένα λήμμα σε ένα συγκεκριμένο φύλλο
17  def Lexicon(lemma, filename='διπλωματικη3.xlsx', sheet_number=7):
18
19      df = pd.read_excel(filename, sheet_name=sheet_number - 1)  # Διαβάζει το φύλλο
20      first_column = df.iloc[:, 0].str.strip()  # Αφαιρεί τα κενά από την πρώτη στήλη
21      sample_lexicon = first_column.tolist()  # Μετατρέπει την πρώτη στήλη σε λίστα
22      results = []  # Δημιουργία κενής λίστας για αποθήκευση των αποτελεσμάτων
23
24      for i, item in enumerate(sample_lexicon):  # Επανάληψη για κάθε λήμμα στο δείγμα
25
26          if item == lemma:  #Αν το λήμμα ταιριάζει
27
28              arithmos = df.iloc[i, 5]  # Λήψη δεδομένων από τη στήλη 6
29              ptosi = df.iloc[i, 10]  # Λήψη δεδομένων από τη στήλη 11
30              onomastiki_enikou_index = df.iloc[i, 2]  # Λήψη δεδομένων από τη στήλη 3
31              meros_logoy_onom_enikoy = df.iloc[i, 4]  # Λήψη δεδομένων από τη στήλη 5
32              genos_onom_enikoy = df.iloc[i, 11]  # Λήψη δεδομένων από τη στήλη 12
33              results.append((lemma, ptosi, arithmos, onomastiki_enikou_index, meros_logoy_onom_enikoy, genos_onom_enikoy))
34
35      if results:
36          return True, results  # Επιστροφή των αποτελεσμάτων αν βρέθηκαν
37      else:
38          return False, None  # Επιστροφή μηδενικών αν δεν βρέθηκαν
```

Στην παραπάνω εικόνα υπάρχουν δύο συναρτήσεις, η πρώτη υπολογίζει το μήκος της

κάθε λέξης και το επιστρέφει στο κύριο πρόγραμμα. Η δεύτερη συνάρτηση αναζητάει την κατάληξη στο φύλλο 7 και επιστρέφει τις πληροφορίες που χρειάζεται το κύριο πρόγραμμα. Μερικές σημαντικές εντολές και διευκρινήσεις παρατίθενται παρακάτω:

- Ο αριθμός του φύλλου μειώνεται κατά 1, επειδή η pandas χρησιμοποιεί μηδενική αρίθμηση.
- df.iloc[:, 0]: Επιλέγει την πρώτη στήλη του DataFrame.
- .str.strip(): Αφαιρεί τα κενά διαστήματα από την αρχή και το τέλος κάθε στοιχείου της στήλης.
- Μετατρέπει τη στήλη σε λίστα χρησιμοποιώντας τη μέθοδο tolist().
- Δημιουργεί μια κενή results για να αποθηκεύσει τα αποτελέσματα της αναζήτησης.
- Ξεκινά έναν βρόχο που επαναλαμβάνει κάθε στοιχείο της λίστας sample_lexicon.
- enumerate(sample_lexicon): Επιστρέφει τόσο το δείκτη (i) όσο και την τιμή (item) κάθε στοιχείου.
- Προσθέτει μια πλειάδα με τις σχετικές πληροφορίες στη results.

```
88  # Συνάρτηση που επιστρέφει πληροφορίες αναφοράς από ένα συγκεκριμένο φύλλο
89  def GetReferenceInfo(onomastiki_enikou_index, filename='διπλωματικη3.xlsx', sheet_number=7):
90      df = pd.read_excel(filename, sheet_name=sheet_number - 1)  # Διαβάζει το φύλλο εργασίας
91      reference_row = df.iloc[onomastiki_enikou_index-2]  # Λήψη της γραμμής αναφοράς
92      return reference_row  # Επιστροφή της γραμμής αναφοράς
93
```

Η συνάρτηση GetReferenceInfo αναζητάει και επιστρέφει τις πληροφορίες μιας συγκεκριμένης γραμμής από ένα φύλλο Excel. Χρησιμοποιεί τη βιβλιοθήκη pandas για να διαβάσει το αρχείο Excel. Στη συνέχεια, χρησιμοποιεί την iloc για να επιλέξει τη γραμμή που αντιστοιχεί στον δεδομένο δείκτη της ονομαστικής ενικού, προσαρμόζοντας τον δείκτη για να ταιριάζει με την αρίθμηση που χρησιμοποιεί η pandas. Η γραμμή με τις πληροφορίες επιστρέφεται στη συνέχεια.

```
102 # Συνάρτηση που μετατρέπει μια λέξη στην ονομαστική ενικού
103 def ConvertToNominative(lexis, postfix, ptosi, arithmos, onomastiki_enikou_indices):
104     nominative_forms = []  # Δημιουργία κενής λίστας για αποθήκευση των ονομαστικών τύπων
105     for index in onomastiki_enikou_indices:  # Επανάληψη για κάθε δείκτη ονομαστικής ενικού
106
107         reference_info = GetReferenceInfo(index)  # Λήψη πληροφοριών αναφοράς
108         nominative_form = reference_info[0]  # Λήψη του ονομαστικού τύπου
109
110         # Αφαίρεση της κατάληξης και προσθήκη της νέας κατάληξης
111         base = lexis[:-len(postfix)]  # Βάση της λέξης χωρίς την κατάληξη
112         new_form = base + nominative_form  # Δημιουργία του νέου τύπου
113
114         nominative_forms.append(new_form)  # Προσθήκη του νέου τύπου στη λίστα
115     return nominative_forms  # Επιστροφή της λίστας ονομαστικών τύπων
116
```

Η συνάρτηση ConvertToNominative μετατρέπει μια λέξη στην ονομαστική ενικού αφαιρώντας την τρέχουσα κατάληξη και προσθέτοντας τη σωστή ονομαστική

κατάληξη από τις πληροφορίες αναφοράς, που βρίσκει στο Excel. Χρησιμοποιεί τους δείκτες των γραμμών (από την στήλη με τον τύπο αναφοράς του Excel), που περιέχουν τις σωστές ονομαστικές καταλήξεις, προσαρμόζει τη λέξη, και επιστρέφει μια λίστα με τους νέους τύπους της λέξης στην ονομαστική ενικού.

```
124
125 # Συνάρτηση που αναλύει αριθμούς από μια συμβολοσειρά
126 def parse_numbers(onomastiki_enikou_index):
127     numbers = re.split(r'[,\s]+', str(onomastiki_enikou_index).strip())  # Διαχωρισμός των αριθμών με βάση τα κόμματα και
128     number_list = []  # Δημιουργία κενής λίστας για αποθήκευση των αριθμών
129     for num in numbers:  # Επανάληψη για κάθε αριθμό στη λίστα
130         if num.isdigit():  # Έλεγχος αν ο αριθμός είναι ψηφίο
131             number_list.append(int(num))  # Προσθήκη του αριθμού στη λίστα ως ακέραιο
132     return number_list  # Επιστροφή της λίστας αριθμών
133
```

Η συνάρτηση parse_numbers λαμβάνει μια συμβολοσειρά που περιέχει αριθμούς διαχωρισμένους με κόμματα ή κενά, διαχωρίζει τη συμβολοσειρά σε ξεχωριστά στοιχεία, ελέγχει αν κάθε στοιχείο είναι ένας έγκυρος ακέραιος αριθμός, και προσθέτει αυτούς τους αριθμούς σε μια λίστα. Στο τέλος, επιστρέφει τη λίστα με τους ακέραιους αριθμούς.

```
140
141 # Συνάρτηση που αναζητάει λέξη σε όλα τα φύλλα (Επαλήθευση)
142 def find_word_in_excel(lexis, filename='διπλωματικη3.xlsx', num_sheets=5):
143     all_results = []  # Δημιουργία κενής λίστας για αποθήκευση όλων των αποτελεσμάτων
144     seen_words = set()  # Δημιουργία για αποθήκευση των ήδη εμφανισμένων λέξεων, πχ τραχύς ειναι δύο φορες
145
146     for sheet_number in range(num_sheets):  # Επανάληψη για κάθε φύλλο
147         df = pd.read_excel(filename, sheet_name=sheet_number)  # Διαβάζει το φύλλο
148
149         for col in df.columns:  # Επανάληψη για κάθε στήλη του φύλλου
150             if df[col].dtype == object:  # Έλεγχος αν ο τύπος των δεδομένων είναι string
151                 if lexis in df[col].values:  # Έλεγχος αν η λέξη υπάρχει στις τιμές της στήλης
152                     result = df.loc[df[col] == lexis]  # Λήψη της γραμμής που περιέχει τη λέξη
153
154                     for _, row in result.iterrows():  # Επανάληψη για κάθε γραμμή στα αποτελέσματα
155                         found_word = row[col]  # Λήψη της λέξης από τη στήλη
156                         if found_word not in seen_words:  # Έλεγχος αν η λέξη δεν έχει ήδη εμφανιστεί
157                             nominative_singular = None  # Δημιουργία μεταβλητής για την ονομαστική ενικού
158
159                             if sheet_number == 0:  # Αν το φύλλο είναι το πρώτο (αρθρα)
160                                 print(row.iloc[0])  # Εκτύπωση της πρώτης στήλης του αποτελέσματος
161
162                             elif sheet_number in [1, 2]:  # Αν το φύλλο είναι το δεύτερο ή το τρίτο (ουσιαστικα, επιθετα &
163                                 result_column3_sheet2_or_3 = row.iloc[2]  # Λήψη της τιμής από την τρίτη στήλη
164                                 nominative_singular = row.iloc[3]  # Λήψη της τιμής από την τέταρτη στήλη
165
166                             elif sheet_number == 4:  # Αν το φύλλο είναι το πέμπτο (ρήματα)
167                                 print(row.iloc[[0, 1]])  # Εκτύπωση της πρώτης και δεύτερης στήλης του αποτελέσματος
168                                 nominative_singular = row.iloc[4]  # Λήψη της τιμής από την πέμπτη στήλη
169
170                             all_results.append((result, result_column3_sheet2_or_3, found_word, nominative_singular))  # Π
171                             seen_words.add(found_word)  # Προσθήκη της λέξης στο σύνολο των ήδη εμφανισμένων λέξεων
172
173     if all_results:  # Έλεγχος αν υπάρχουν αποτελέσματα
174         return all_results  # Επιστροφή των αποτελεσμάτων
175     else:
176         return None  # Επιστροφή μηδενικών αν δεν υπάρχουν αποτελέσματα
177
```

Η συνάρτηση αναζητά μια συγκεκριμένη λέξη σε όλα τα φύλλα. Κάθε φύλλο διαβάζεται και ελέγχει κάθε στήλη για την ύπαρξη της λέξης. Αν βρεθεί η λέξη, τα αντίστοιχα δεδομένα καταχωρούνται σε μια λίστα αποτελεσμάτων, αποφεύγοντας την καταχώριση διπλότυπων λέξεων. Στο τέλος, επιστρέφει τη λίστα με τα αποτελέσματα ή None αν δεν βρέθηκαν τα αποτελέσματα.

```python
def search_word_and_get_columns(filename, sheet_number , word):
    # Διαβάζουμε το 5ο φύλλο του αρχείου Excel
    df = pd.read_excel(filename, sheet_name=sheet_number - 1)

    # Αναζητάμε τη λέξη σε όλες τις στήλες του φύλλου
    result = df.apply(lambda row: row.astype(str).str.contains(word).any(), axis=1)

    # Φιλτράρουμε τις γραμμές που περιέχουν τη λέξη
    rows_with_word = df[result]

    # Αποθηκεύουμε τις τιμές από την 1η, 2η και 3η στήλη
    results2 = []

    for idx, row in rows_with_word.iterrows():

        voice = row.iloc[0]
        tense = row.iloc[1]
        mood = row.iloc[2]

        line_number = idx + 2  # Προσθέτουμε 2 επειδή οι γραμμές στο Excel ξεκινούν από το 2 (1-indexed)
        if tense != 'ενεστώτας' and mood != 'οριστ. /υποτ.' :

            for i in range(line_number, -1, -1):
                if df.iloc[i, 1] == "ενεστώτας" and df.iloc[i, 2] == "οριστ. /υποτ.":
                    lexis = df.iloc[i, 5]

                    results2.append((lexis, voice, tense, mood, line_number))
                    break
        elif tense == 'ενεστώτας' and mood == 'οριστ. /υποτ.' :
            for i in range(line_number, -1, -1):
                if df.iloc[i, 1] == "ενεστώτας" and df.iloc[i, 2] == "οριστ. /υποτ.":
                    lexis = df.iloc[i, 5]

                    results2.append((lexis, voice, tense, mood, line_number))
                    break
        return results2
    return None
```

Αυτό το κομμάτι του κώδικα αναζητάει τα ρήματα που βρίσκονται στο 5ο φύλλο του Excel. Τα ρήματα επειδή δεν έχουν καλή διάταξη στο αρχείο, η ονομαστική ενικού της συγκεκριμένης φωνής βρίσκεται με τέχνασμα. Αν το ρήμα είναι σε ονομαστική ενικού, αποθηκεύει απευθείας τις πληροφορίες. Εάν όμως είναι αλλού, τότε επιστρέφει προς τα πίσω στο Excel έως ότου να βρει τον χρόνο ενεστώτα και έγκλιση οριστική και υποτακτική. Ο συγκεκριμένος χρόνος και έγκλιση είναι μοναδικός και τα στοιχεία αποθηκεύονται και επεξεργάζονται αργότερα στο κύριο πρόγραμμα.

```python

# Συνάρτηση που μετατρέπει ένα κείμενο σε πεζά γράμματα και αφαιρεί τόνους
def to_lowercase(text):
    if isinstance(text, float) and pd.isna(text):  # Έλεγχος αν το κείμενο είναι NaN
# Η εντολή isinstance επιστρέφει True αν το TEXT ειναι δεκαδικός, ειναι σε συγκεκριμένο τύπο
        return ""
    if not isinstance(text, str):  # Έλεγχος αν το κείμενο δεν είναι συμβολοσειρά
        text = str(text)  # Μετατροπή του κειμένου σε συμβολοσειρά
    normalized_text = unicodedata.normalize('NFD', text)  # Κανονικοποίηση του κειμένου (αφαίρεση τόνου, γιατι μπορει να μ
    lowercase_text = normalized_text.lower()  # Μετατροπή του κειμένου σε πεζά γράμματα
    result_text = unicodedata.normalize('NFC', lowercase_text)  # Επαναφορά της κανονικοποίησης
    return result_text  # Επιστροφή του αποτελέσματος

```

Η συνάρτηση μετατρέπει το κείμενο σε πεζά γράμματα και αφαιρεί τους τόνους. Ελέγχει αν το κείμενο είναι NaN ή αν δεν είναι ήδη, το μετατρέπει αναλόγως. Στη συνέχεια, κανονικοποιεί το κείμενο, το μετατρέπει σε πεζά γράμματα και το επανακανονικοποιεί, πριν επιστρέψει το αποτέλεσμα.

```python
# Ορισμός της λίστας με τα άρθρα και τις λεπτομέρειές τους
articles_list = [
    {'Article': 'ο', 'Gender': 'αρσενικό', 'Number': 'ενικός', 'Case': 'ονομαστική'},
    {'Article': 'του', 'Gender': 'αρσενικό', 'Number': 'ενικός', 'Case': 'γενική'},
    {'Article': 'τον', 'Gender': 'αρσενικό', 'Number': 'ενικός', 'Case': 'αιτιατική'},
    {'Article': 'οι', 'Gender': 'αρσενικό', 'Number': 'πληθυντικός', 'Case': 'ονομαστική'},
    {'Article': 'των', 'Gender': 'αρσενικό', 'Number': 'πληθυντικός', 'Case': 'γενική'},
    {'Article': 'τους', 'Gender': 'αρσενικό', 'Number': 'πληθυντικός', 'Case': 'αιτιατική'},
    {'Article': 'η', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'ονομαστική'},
    {'Article': 'της', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'γενική'},
    {'Article': 'την', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'αιτιατική'},
    {'Article': 'οι', 'Gender': 'θηλυκό', 'Number': 'πληθυντικός', 'Case': 'ονομαστική'},
    {'Article': 'των', 'Gender': 'θηλυκό', 'Number': 'πληθυντικός', 'Case': 'γενική'},
    {'Article': 'τις', 'Gender': 'θηλυκό', 'Number': 'πληθυντικός', 'Case': 'αιτιατική'},
    {'Article': 'το', 'Gender': 'ουδέτερο', 'Number': 'ενικός', 'Case': 'ονομαστική/αιτιατική'},
    {'Article': 'του', 'Gender': 'ουδέτερο', 'Number': 'ενικός', 'Case': 'γενική'},
    {'Article': 'τα', 'Gender': 'ουδέτερο', 'Number': 'πληθυντικός', 'Case': 'ονομαστική/αιτιατική'},
    {'Article': 'των', 'Gender': 'ουδέτερο', 'Number': 'πληθυντικός', 'Case': 'γενική'},
    # Προσθέστε όσα άρθρα και λεπτομέρειες χρειάζονται
]

def find_article_details(article_to_find):
    for index, article_info in enumerate(articles_list):
        if article_info['Article'] == article_to_find:
            return {
                'Article': article_info['Article'],
                'Gender': article_info['Gender'],
                'Number': article_info['Number'],
                'Case': article_info['Case'],

            }
    return None
```

Σε αυτό το κομμάτι του κώδικα εκχωρούμε σε μια λίστα όλα τα άρθρα και τις πληροφορίες του για να τα χρησιμοποιήσουμε στην συνάρτηση της εικόνας. Αυτή η συνάρτηση είναι απαραίτητη, καθώς λόγω της LoMaFi κόβει το άρθρο για να βρει κατάληξη. Έτσι ψάχνει καθαυτού το άρθρο, χωρίς να το κόβει στο κύριο πρόγραμμα.

### Κύριο πρόγραμμα

```python
# Κύριο πρόγραμμα


lexis = "φως"

results = Find_Postfix(lexis)


article_details = find_article_details(lexis)

filename = 'διπλωματικη3.xlsx'
sheet_number = 5
results2 = search_word_and_get_columns(filename, sheet_number, lexis)


if article_details:
    print('Η λέξη είναι άρθρο!')
    print(f"Άρθρο   : {article_details['Article']}")
    print(f"Γένος   : {article_details['Gender']}")
    print(f"Αριθμός : {article_details['Number']}")
    print(f"Πτώση   : {article_details['Case']}")

elif article_details != None:
    print(f"Το άρθρο '{lexis}' δεν βρέθηκε.")

elif results2:

    for lexis, voice, tense, mood, line_number in results2:
        print('Η λέξη είναι ρήμα!')
        print ('Η Γραμμή της λέξης', lexis, ':', line_number)
        print('Ο Ενεστώτας οριστικης ειναι :', lexis)
        print('Η Φωνη της λέξης είναι      :', voice)
        print('Ο Χρονος της λέξης είναι    :', tense)
        print('Η Εγκλιση της λέξης είναι   :', mood)
```

Στο κύριο πρόγραμμα υπάρχει μια μεταβλητή με το όνομα lexis. Στην παραπάνω

εικόνα η λέξη αυτή είναι το “φως”, αναζητάει την λέξη φως στην βάση δεδομένων και εμφανίζει τις πληροφορίες της. Ύστερα καλεί τις συναρτήσεις που χρειάζεται για να βρει αν είναι άρθρο ή ρήμα. Πρώτα ψάχνει ολόκληρη την λέξη, αν είναι άρθρο χωρίς να αφαιρεί γράμματα από την LoMaFi και αν βρει ότι είναι άρθρο εκτυπώνει τις λοιπές πληροφορίες της. Μετά ελέγχει αν είναι ρήμα επίσης με τον ίδιο τρόπο.

```
elif results or article_details == None and results2 == None  :  # Έλεγχος αν υπάρχουν αποτελέσματα
    print('Η λέξη είναι ουσιαστικό και επίθετο!')
    for i, result in enumerate(results, start=1):  # Επανάληψη για κάθε αποτέλεσμα

        postfix, ptosi, arithmos, onomastiki_enikou_index, meros_logoy_onom_enikoy, genos_onom_enikoy = result
        ar = parse_numbers(onomastiki_enikou_index)  # Ανάλυση των αριθμών
        count = len(ar)  # Υπολογισμός του πλήθους των αριθμών
        print(f"Αποτέλεσμα {i}      :")  # Εκτύπωση του αποτελέσματος
        print("Βρέθηκε η κατάληξη:", postfix)  # Εκτύπωση της κατάληξης
        print("Η πτώση είναι     :", ptosi)  # Εκτύπωση της πτώσης
        print("Ο αριθμός είναι   :", arithmos)  # Εκτύπωση του αριθμού

        if count > 1:  # Έλεγχος αν ο αριθμός των αναφορών είναι μεγαλύτερος του 1

            print(f"Συνολικός αριθμός αναφορών: {count}")  # Εκτύπωση του συνολικού αριθμού αναφορών

            for idx in ar:  # Επανάληψη για κάθε δείκτη αναφοράς

                reference_info = GetReferenceInfo(idx)  # Λήψη πληροφοριών αναφοράς
                print(f"Αναφορά {idx}       :")  # Εκτύπωση της αναφοράς
                print('Η κατάληξη του    :', reference_info[0])  # Εκτύπωση της κατάληξης


        elif count == 1 :  # Έλεγχος αν ο αριθμός των αναφορών είναι 1

            reference_info = GetReferenceInfo(ar[0])  # Λήψη πληροφοριών αναφοράς
            print('Η κατάληξη του    :', reference_info[0])  # Εκτύπωση της κατάληξης
            print("Μέρος του λόγου   :", reference_info[4])  # Εκτύπωση του μέρους του λόγου
            print("Το γένος είναι    :", reference_info[11])  # Εκτύπωση του γένους
```

Έπειτα αν δεν είναι ούτε άρθρο, ούτε ρήμα ψάχνει για κατάληξη καλώντας τις συναρτήσεις. Αρχικά όταν βρίσκει την κατάληξη, αναζητάει αν υπάρχουν ίδιες καταλήξεις για να βρει την σωστή. Παρακάτω μαζί με τις καταλήξεις εμφανίζει και τον τύπο αναφοράς, οπού μπορεί να είναι ένας η παραπάνω αριθμοί. Αν ο τύπος αναφοράς είναι μεγαλύτερος του ένα, κάνει επανάληψη για όλες τις καταλήξεις ονομαστικής ενικού. Εάν είναι ένας εκτυπώνει τις πληροφορίες του.

```
            print("Μέρος του λόγου   :", reference_info[4])  # Εκτύπωση του μέρους του λόγου
            print("Το γένος είναι    :", reference_info[11])  # Εκτύπωση του γένους

        k = 0
        l = 0
        excel_results = find_word_in_excel(lexis)  # Αναζήτηση της λέξης στο Excel

        if excel_results:  # Έλεγχος αν υπάρχουν αποτελέσματα

            for excel_result in excel_results:  # Επανάληψη για κάθε αποτέλεσμα

                result, result_column3, found_word, nominative_singular = excel_result  # Λήψη των αποτελεσμάτων
                lowercase_text = to_lowercase(genos_onom_enikoy)  # Μετατροπή του γένους σε πεζά γράμματα
                result_column3_lower = to_lowercase(result_column3)  # Μετατροπή του τρίτου αποτελέσματος σε πεζά γράμματα

                if result_column3_lower == genos_onom_enikoy:  # Έλεγχος αν τα γένη ταιριάζουν

                    if lexis == found_word:  # Έλεγχος αν η λέξη βρέθηκε

                        lexis_nominative = ConvertToNominative(lexis, postfix, ptosi, arithmos, ar)  # Μετατροπή της λέξης

                        for nom in lexis_nominative:  # Επανάληψη για κάθε ονομαστική μορφή

                            k = k + 1

                            if nominative_singular == nom and k == 1:  # Έλεγχος αν η ονομαστική μορφή ταιριάζει

                                print("Μέρος του λόγου   :", meros_logoy_onom_enikoy)  # Εκτύπωση του μέρους του λόγου
                                print("Το γένος είναι    :", genos_onom_enikoy)  # Εκτύπωση του γένους
                                print("Η λέξη στην ονομαστική ενικού είναι:", nom)  # Εκτύπωση της λέξης στην ονομαστική ε
```

Παρακάτω κάνει αναζήτηση της λέξης, εάν υπάρχει η λέξη στην βάση δεδομένων μετατρέπει την λέξη στην ονομαστική ενικού και κάνει επαλήθευση ότι υπάρχει τέτοια λέξη.

```
367                 elif lexis != found_word:  # Έλεγχος αν η λέξη δεν βρέθηκε
368
369                     print('Η λέξη δεν υπάρχει')  # Εκτύπωση μηνύματος ότι η λέξη δεν υπάρχει
370
371                 l = l + 1
372
373                 if ptosi == 'ονομαστική' and arithmos == 'ενικός' and l == 1:  # Έλεγχος αν η πτώση και ο αριθμός ταιρ
374
375                     print("Η λέξη στην ονομαστική ενικού είναι:", lexis)  # Εκτύπωση της λέξης στην ονομαστική ενικού
376
377             if result_column3_lower != genos_onom_enikoy:  # Έλεγχος αν τα γένη δεν ταιριάζουν
378
379                 print(result_column3_lower, 'vs', genos_onom_enikoy)  # σύγκριση στα γένη
380                 print('ΔΕΝ ΤΑΙΡΙΑΖΕΙ ΤΟ ΓΕΝΟΣ')  # Εκτύπωση μηνύματος ότι τα γένη δεν ταιριάζουν
381
382 elif results == None or article_details == None and results2 == None:
383     print("Δεν βρέθηκαν πληροφορίες για την λέξη.")  # Εκτύπωση μηνύματος αν δεν βρέθηκαν πληροφορίες για την κατάληξη
```

Επιπλέον αν η λέξη δεν υπάρχει, εκτυπώνει το κατάλληλο μήνυμα, ενώ αν η λέξη υπάρχει και είναι ήδη στην ονομαστική, την εμφανίζει όπως είναι. Μετά ελέγχει αν τα γένη είναι ίδια, ένας άλλος τρόπος επαλήθευσης για να εμφανιστεί το σωστό αποτέλεσμα. Τέλος ελέγχει αν δεν είναι άρθρο, ρήμα ή ουσιαστικό και επίθετο να εμφανίζει ότι δεν βρέθηκαν οι πληροφορίες.

Τα αποτελέσματα από την λέξη *φως* παρατίθενται παρακάτω:

```
Console 1/A
Βρέθηκε η κατάληξη: ως
Η πτώση είναι     : ονομαστική
Ο αριθμός είναι   : ενικός
Η λέξη στην ονομαστική ενικού είναι: φως
Αποτέλεσμα 2      :
Βρέθηκε η κατάληξη: ως
Η πτώση είναι     : γενικής
Ο αριθμός είναι   : ενικός
Η κατάληξη του    : ω
Μέρος του λόγου   : ουσιαστικό και επίθετο
Το γένος είναι    : θηλυκό
ουδέτερο vs θηλυκό
ΔΕΝ ΤΑΙΡΙΑΖΕΙ ΤΟ ΓΕΝΟΣ
Αποτέλεσμα 3      :
Βρέθηκε η κατάληξη: ως
Η πτώση είναι     : αιτιατική
Ο αριθμός είναι   : ενικός
Η κατάληξη του    : ως
Μέρος του λόγου   : ουσιαστικό και επίθετο
Το γένος είναι    : ουδέτερο
Μέρος του λόγου   : ουσιαστικό και επίθετο
Το γένος είναι    : ουδέτερο
Η λέξη στην ονομαστική ενικού είναι: φως
Αποτέλεσμα 4      :
Βρέθηκε η κατάληξη: ως
Η πτώση είναι     : κλητική
Ο αριθμός είναι   : ενικός
Η κατάληξη του    : ως
Μέρος του λόγου   : ουσιαστικό και επίθετο
Το γένος είναι    : ουδέτερο
Μέρος του λόγου   : ουσιαστικό και επίθετο
Το γένος είναι    : ουδέτερο
Η λέξη στην ονομαστική ενικού είναι: φως
```

| | | |
|---|---|---|
| 6 | ως | Καταλήξεις ονομαστικής ενικού αριθμού ουδετέρων ουσιαστικών και επιθέτων |
| 7 | ως | Καταλήξεις γενικής ενικού αριθμού θηλυκών ουσιαστικών και επιθέτων |
| 8 | ως | Καταλήξεις αιτιατικής ενικού αριθμού ουδετέρων ουσιαστικών και επιθέτων |
| 9 | ως | Καταλήξεις κλητικής ενικού αριθμού ουδετέρων ουσιαστικών και επιθέτων |
| 0 | ώς | Καταλήξεις ονομαστικής ενικού αριθμού ουδετέρων ουσιαστικών και επιθέτων |

## 4.3 Κώδικας 2

Ο κώδικας 2 είναι ένα άλλο κομμάτι από αυτό των καταλήξεων. Σε αυτόν τον κώδικα θα βρίσκει από τις προτάσεις το υποκείμενο, ρήμα και αντικείμενα. Επίσης να σημειωθεί πως όταν τα βρίσκει, τα απαριθμεί σαν την Relational grammar. Για την καλύτερη κατανόηση του κώδικα θα αναλυθεί τμήμα - τμήμα παρακάτω:

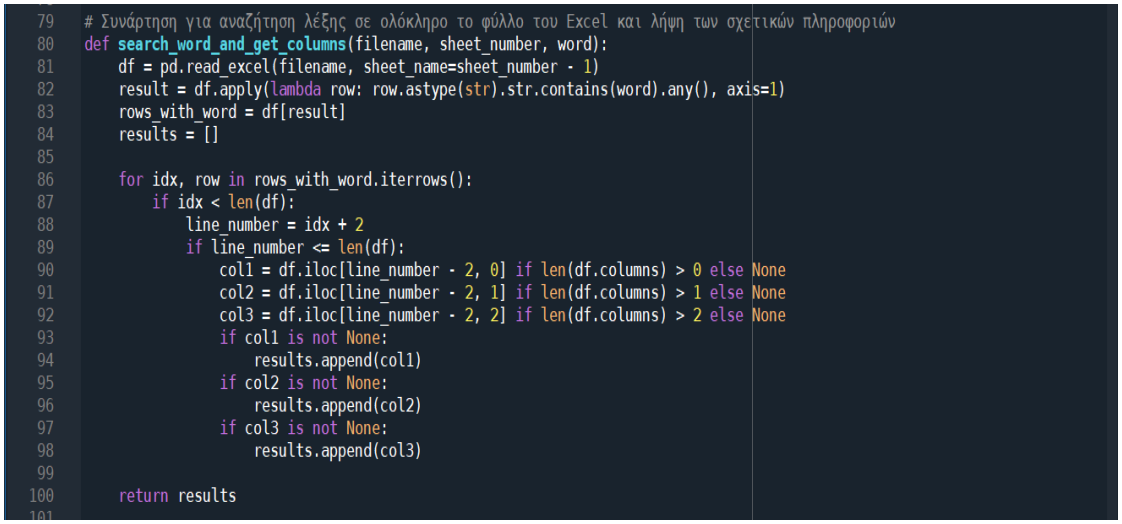
**Συναρτήσεις**

```
# Λίστα με τα άρθρα και τις λεπτομέρειές τους
articles_list = [
    {'Article': 'ο', 'Gender': 'αρσενικό', 'Number': 'ενικός', 'Case': 'ονομαστική'},
    {'Article': 'του', 'Gender': 'αρσενικό', 'Number': 'ενικός', 'Case': 'γενική'},
    {'Article': 'τον', 'Gender': 'αρσενικό', 'Number': 'ενικός', 'Case': 'αιτιατική'},
    {'Article': 'οι', 'Gender': 'αρσενικό', 'Number': 'πληθυντικός', 'Case': 'ονομαστική'},
    {'Article': 'των', 'Gender': 'αρσενικό', 'Number': 'πληθυντικός', 'Case': 'γενική'},
    {'Article': 'τους', 'Gender': 'αρσενικό', 'Number': 'πληθυντικός', 'Case': 'αιτιατική'}
    {'Article': 'η', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'ονομαστική'},
    {'Article': 'της', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'γενική'},
    {'Article': 'την', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'αιτιατική'},
    {'Article': 'οι', 'Gender': 'θηλυκό', 'Number': 'πληθυντικός', 'Case': 'ονομαστική'},
    {'Article': 'των', 'Gender': 'θηλυκό', 'Number': 'πληθυντικός', 'Case': 'γενική'},
    {'Article': 'τις', 'Gender': 'θηλυκό', 'Number': 'πληθυντικός', 'Case': 'αιτιατική'},
    {'Article': 'το', 'Gender': 'ουδέτερο', 'Number': 'ενικός', 'Case': 'ονομαστική/αιτιατ
    {'Article': 'του', 'Gender': 'ουδέτερο', 'Number': 'ενικός', 'Case': 'γενική'},
    {'Article': 'τα', 'Gender': 'ουδέτερο', 'Number': 'πληθυντικός', 'Case': 'ονομαστική/α
    {'Article': 'των', 'Gender': 'ουδέτερο', 'Number': 'πληθυντικός', 'Case': 'γενική'},
    {'Article': 'στου', 'Gender': 'αρσενικό', 'Number': 'ενικός', 'Case': 'γενική'},
    {'Article': 'στον', 'Gender': 'αρσενικό', 'Number': 'ενικός', 'Case': 'αιτιατική'},
    {'Article': 'στων', 'Gender': 'αρσενικό', 'Number': 'πληθυντικός', 'Case': 'γενική'},
    {'Article': 'στους', 'Gender': 'αρσενικό', 'Number': 'πληθυντικός', 'Case': 'αιτιατική'
    {'Article': 'τη', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'αιτιατική'},
    {'Article': 'στης', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'γενική'},
    {'Article': 'στην', 'Gender': 'θηλυκό', 'Number': 'ενικός', 'Case': 'αιτιατική'},
    {'Article': 'στων', 'Gender': 'θηλυκό', 'Number': 'πληθυντικός', 'Case': 'γενική'},
    {'Article': 'στις', 'Gender': 'θηλυκό', 'Number': 'πληθυντικός', 'Case': 'αιτιατική'}

]


# Συνάρτηση για εύρεση λεπτομερειών άρθρου
def find_article_details(article_to_find):
    for article_info in articles_list:
        if article_info['Article'].lower() == article_to_find.lower():
            return article_info
    return None
```

Στην παραπάνω εικόνα υπάρχει η λίστα με όλα τα άρθρα και τις πληροφορίες τους. Μερικές από αυτές είναι το γένος, αριθμός και πτώση. Στην συνέχεια υπάρχει μια συνάρτηση η οποία με βάση το άρθρο, εμφανίζει τις κατάλληλες πληροφορίες. Αυτό γίνεται με μία σύγκριση στην if, χωρίς να λαμβάνει υπόψιν τα κεφαλαία και πεζά γράμματα.

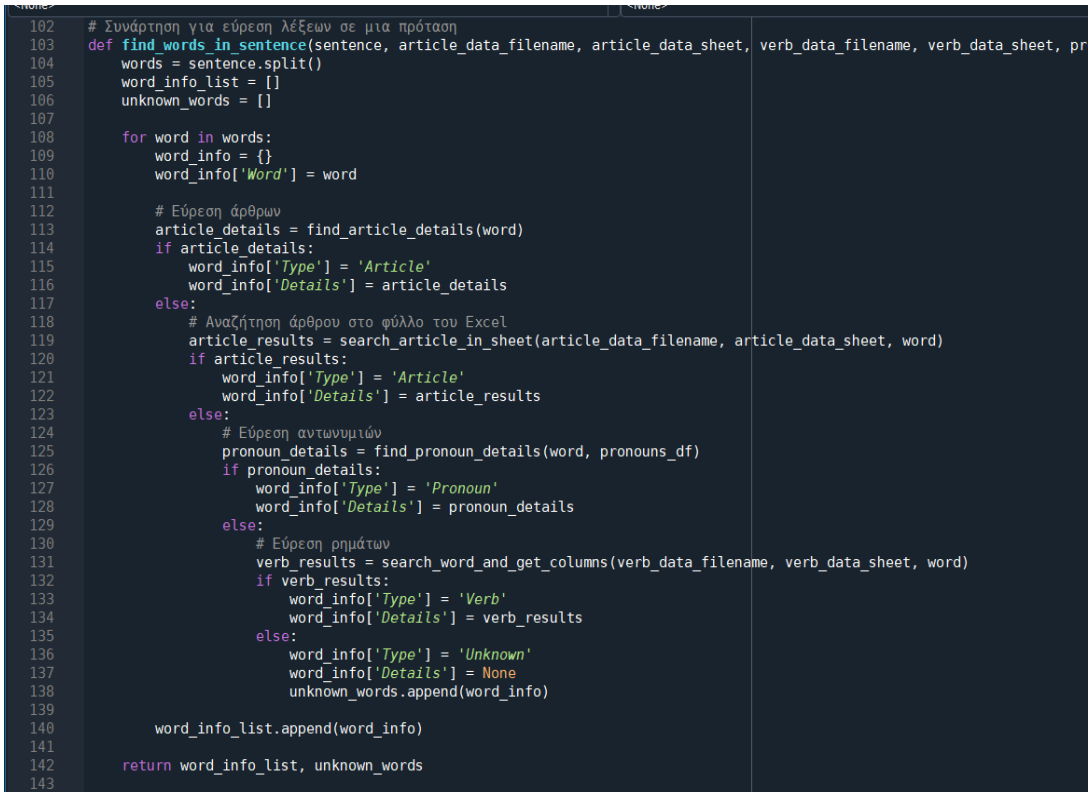
```
# Συνάρτηση για εύρεση αντωνυμιών και των λεπτομερειών τους
def find_pronoun_details(pronoun_to_find, pronouns_df):
    for col in pronouns_df.columns:
        if pronouns_df[col].astype(str).str.contains(pronoun_to_find, case=False, na=False).any():
            pronoun_row = pronouns_df[pronouns_df[col].astype(str).str.contains(pronoun_to_find, case=False, na=False)]
            details = []
            if not pronoun_row.empty:
                if len(pronouns_df) > 0:
                    details.append(pronouns_df.iloc[0, pronouns_df.columns.get_loc(col)])
                if len(pronouns_df) > 1:
                    details.append(pronouns_df.iloc[1, pronouns_df.columns.get_loc(col)])
            return details
    return None
```

Αυτή η συνάρτηση ψάχνει για μια συγκεκριμένη αντωνυμία μέσα σε ένα DataFrame. Για να γίνει όλη αυτή η διαδικασία ψάχνει σε κάθε στήλη την αντωνυμία που αναζητείται. Η διαδικασία υλοποιείται με την μέθοδο “contains”, η οποία ελέγχει για την ύπαρξη της αντωνυμίας με τις εντολές “ iloc” και “get_loc”. Αυτές οι εντολές επιστρέφουν τις σχετικές λεπτομέρειες, όπως οι δύο πρώτες στήλες με τις πληροφορίες της. Εάν δεν βρεθεί η συγκεκριμένη συνάρτηση, εμφανίζει None.



```python
# Συνάρτηση για αναζήτηση λέξης σε ολόκληρο το φύλλο του Excel και λήψη των σχετικών πληροφοριών
def search_word_and_get_columns(filename, sheet_number, word):
    df = pd.read_excel(filename, sheet_name=sheet_number - 1)
    result = df.apply(lambda row: row.astype(str).str.contains(word).any(), axis=1)
    rows_with_word = df[result]
    results = []

    for idx, row in rows_with_word.iterrows():
        if idx < len(df):
            line_number = idx + 2
            if line_number <= len(df):
                col1 = df.iloc[line_number - 2, 0] if len(df.columns) > 0 else None
                col2 = df.iloc[line_number - 2, 1] if len(df.columns) > 1 else None
                col3 = df.iloc[line_number - 2, 2] if len(df.columns) > 2 else None
                if col1 is not None:
                    results.append(col1)
                if col2 is not None:
                    results.append(col2)
                if col3 is not None:
                    results.append(col3)

    return results
```

Σε αυτή την εικόνα η συνάρτηση αναζητάει μια συγκεκριμένη λέξη σε ένα ολόκληρο φύλλο Excel που έχει καθοριστεί στο κύριο πρόγραμμα. Στη συνέχεια με την μέθοδο apply  ελέγχει κάθε σειρά αν υπάρχει η λέξη που αναζητείται. Εφόσον βρεθεί η λέξη, αποθηκεύει τις τιμές των πρώτων τριών στηλών της αντίστοιχης γραμμής, αν δεν βρεθεί, τότε επιστέφει None.



```python
# Συνάρτηση για εύρεση λέξεων σε μια πρόταση
def find_words_in_sentence(sentence, article_data_filename, article_data_sheet, verb_data_filename, verb_data_sheet, pr
    words = sentence.split()
    word_info_list = []
    unknown_words = []

    for word in words:
        word_info = {}
        word_info['Word'] = word

        # Εύρεση άρθρων
        article_details = find_article_details(word)
        if article_details:
            word_info['Type'] = 'Article'
            word_info['Details'] = article_details
        else:
            # Αναζήτηση άρθρου στο φύλλο του Excel
            article_results = search_article_in_sheet(article_data_filename, article_data_sheet, word)
            if article_results:
                word_info['Type'] = 'Article'
                word_info['Details'] = article_results
            else:
                # Εύρεση αντωνυμιών
                pronoun_details = find_pronoun_details(word, pronouns_df)
                if pronoun_details:
                    word_info['Type'] = 'Pronoun'
                    word_info['Details'] = pronoun_details
                else:
                    # Εύρεση ρημάτων
                    verb_results = search_word_and_get_columns(verb_data_filename, verb_data_sheet, word)
                    if verb_results:
                        word_info['Type'] = 'Verb'
                        word_info['Details'] = verb_results
                    else:
                        word_info['Type'] = 'Unknown'
                        word_info['Details'] = None
                        unknown_words.append(word_info)

        word_info_list.append(word_info)

    return word_info_list, unknown_words
```

Η find_words_in_sentence αναλύει την πρόταση σε λέξεις και βρίσκει πληροφορίες για κάθε λέξη, όπως αν είναι άρθρο, αντωνυμία ή ρήμα. Έπειτα καλεί τις κατάλληλες συναρτήσεις για να βρει αν είναι άρθρο, αντωνυμία ή ρήμα. Όταν βρίσκει τι είναι τι, επιστρέφει τις κατάλληλες πληροφορίες. Αν δεν βρεθεί τι είναι η κάθε λέξη, επιστρέφει 'Unknown' και στις πληροφορίες 'None'.

```
143
144 # Συνάρτηση για διάβασμα των προτάσεων και εύρεση λέξεων
145 def read_excel_and_find_words(sentences_filename, sentences_sheet_number, article_data_filename, article_data_sheet_number, verb_data_f
146     sentences_df = pd.read_excel(sentences_filename, sheet_name=sentences_sheet_number - 1)
147     sentences_info = []
148     all_unknown_words = []
149
150     for index, row in sentences_df.iterrows():
151         words = [str(word) for word in row if pd.notnull(word)]
152         sentence = ' '.join(words)
153         word_info_list, unknown_words = find_words_in_sentence(sentence, article_data_filename, article_data_sheet_number, verb_data_fi
154         sentences_info.append({'sentence': sentence, 'word_info_list': word_info_list})
155         all_unknown_words.extend(unknown_words)
156
157     return sentences_info, all_unknown_words
158
159 # Συνάρτηση για αναζήτηση λέξεων σε άλλο φύλλο του Excel και εκτύπωση των πληροφοριών τους
160 def search_words_and_print_info(filename, words_sheet_number, unknown_words):
161     words_df = pd.read_excel(filename, sheet_name=words_sheet_number - 1)
162
163     for word_info in unknown_words:
164         word = word_info['Word']
165         word_row = words_df[words_df.apply(lambda x: x.astype(str).str.contains(word, case=False, na=False).any(), axis=1)]
166         if not word_row.empty:
167             row_idx = word_row.index[0]
168             col_idx = word_row.columns.get_loc(word_row.apply(lambda x: x.astype(str).str.contains(word, case=False, na=False).idxmax()
169
170             first_row_value = words_df.columns[col_idx]  # Επικεφαλίδα στήλης
171             first_col_value = words_df.iloc[row_idx, 0] if len(words_df.columns) > 0 else None
172             second_col_value = words_df.iloc[row_idx, 1] if len(words_df.columns) > 1 else None
173             arithmos = words_df.iloc[0, col_idx ] if len(words_df.columns) > 2 else None
174             word_info['meros'] = first_col_value
175             word_info['ptosi'] = first_row_value
176             word_info['genos'] = second_col_value
177             word_info['arithmos'] = arithmos
178     return unknown_words
179
```

Η συνάρτηση read_excel_and_find_words διαβάζει προτάσεις από ένα αρχείο Excel και αναλύει τις λέξεις της. Συγκεκριμένα χρησιμοποιείται η βιβλιοθήκη Pandas, η οποία διαβάζει τα δεδομένα από το Excel. Ο προσδιορισμός του Excel γίνεται με το όνομα του αρχείου (Sentences_filename) και τον αριθμό του φύλλου (sentences_sheet_number). Όλα τα δεδομένα αποθηκεύονται σε ένα DataFrame (sentences_df).

Στη συνέχεια διατρέχει κάθε γραμμή συνενώνοντας κάθε λέξη της γραμμής, σχηματίζοντας μια πρόταση. Παράλληλα καλεί την συνάρτηση find_words_in_sentence, όπου αναλύει τις λέξεις κάθε πρότασης και επιστρέφει πληροφορίες για κάθε λέξη. Οι πληροφορίες αυτές υποδεικνύουν αν είναι άρθρο, αντωνυμία ή ρήμα, και αυτές οι πληροφορίες αποθηκεύονται σε μια λίστα (sentences_info). Οι λέξεις που δεν αναγνωρίζονται, καταγράφονται σε μια άλλη λίστα η οποία ονομάζεται all_unknown_words.

Παρακάτω η συνάρτηση Search_words_and_print_info παράγει τα αποτελέσματα των λέξεων που αρχικά δεν βρέθηκαν και εκτυπώνει τις πληροφορίες τους. Η επεξεργασία αυτή γίνεται πάλι με την βιβλιοθήκη Pandas, για να διαβάσει τα δεδομένα από το Excel που προσδιορίζεται ως filename και τον αριθμό του φύλλου ως words_sheet_number.

Για τις λέξεις που δεν αναγνωρίστηκαν και καταγράφηκαν στην λίστα unknown_words, η συνάρτηση ψάχνει εκ νέου σε ένα άλλο Excel, όπου αρχικά εντοπίζει την σειρά και την στήλη που βρέθηκε και ύστερα καταγράφει τις λοιπές πληροφορίες. Όλες οι πληροφορίες που χρειάζονται στο κύριο πρόγραμμα

αποθηκεύονται στην αρχική λίστα unknown_words, όπου προσθέτει τις πληροφορίες που δεν εντοπίστηκαν αρχικά.

```
179
180 def remove_subject_and_verb(sentence, subject, verb):
181     # Διαχωρισμός της πρότασης σε λέξεις
182     words = sentence.split()
183
184     # Αφαίρεση του υποκειμένου
185     subject_words = subject.split()
186     remaining_words = [word for word in words if word not in subject_words]
187
188     # Αφαίρεση του ρήματος
189     remaining_words = [word for word in remaining_words if word != verb]
190
191     # Επιστροφή της υπόλοιπης πρότασης
192     remaining_sentence = ' '.join(remaining_words)
193     return remaining_sentence
194
195 def count_words(remaining_sentence):
196     # Χωρίζουμε την πρόταση σε λέξεις με βάση τα κενά διαστήματα
197     words = remaining_sentence.split()
198     # Επιστρέφουμε τον αριθμό των λέξεων
199     return len(words)
200 def find_article_position(words_list):
201     # Λίστα με τα ελληνικά άρθρα
202     articles = ['ο', 'του', 'τον', 'οι', 'των', 'τους', 'η', 'της', 'την', 'το', 'τα', 'στου
203
204     # Έλεγχος αν κάποια από τις λέξεις είναι άρθρο και επιστροφή της θέσης του
205     for index, word in enumerate(words_list):
206         if word in articles:
207             return index
208
209     # Αν δεν βρεθεί άρθρο, επιστρέφουμε -1
210     return -1
211
```

Η συνάρτηση remove_subject_and_verb αφαιρεί το υποκείμενο και το ρήμα και επιστρέφει την υπόλοιπη πρόταση. Έπειτα η συνάρτηση διαχωρίζει την πρόταση σε λέξεις χρησιμοποιώντας τα κενά διαστήματα. Στην συνέχεια αφαιρεί το υποκείμενο το οποίο βρίσκεται στην ονομαστική πτώση, ένα υποκείμενο καθορίζεται με μια ή περισσότερες λέξεις (Ο γιατρός, η μαύρη γάτα κτλ.) οπότε αφαιρούνται όλες. Επίσης αφαιρεί το ρήμα που εύκολα εντοπίζεται από τις καταλήξεις και ότι απομείνει το συνενώνει σε μια ενιαία πρόταση και την επιστρέφει.

Η count_words διαχωρίζει όλες τις λέξεις της πρότασης ανάλογα με τα κενά διαστήματα και μετράει τον αριθμό των λέξεων και τον επιστρέφει. Ουσιαστικά, η συνάρτηση αυτή είναι ένας απλός μετρητής λέξεων.

Τέλος η συνάρτηση find_article_position αναζητά την θέση του άρθρου και επιστρέφει τον δείκτη της. Μετά αρχικοποιεί μια λίστα με όλα τα άρθρα και από αυτήν αναζητά αν υπάρχει το άρθρο. Αν βρεθεί κάποια λέξη η οποία είναι άρθρο τότε επιστρέφει την θέση του. Άμα δεν βρεθεί τότε εμφανίζει -1.

**Κύριο πρόγραμμα**

```
212
213  # Παράδειγμα χρήσης των συναρτήσεων
214  sentences_filename = 'προτασεις.xlsx'
215  sentences_sheet_number = 1
216  article_data_filename = 'διπλωματικη3.xlsx'
217  article_data_sheet_number = 1
218  verb_data_filename = 'διπλωματικη3.xlsx'
219  verb_data_sheet_number = 5
220  pronouns_sheet_number = 4
221  words_sheet_number = 2
222
223  pronouns_df = pd.read_excel(verb_data_filename, sheet_name=pronouns_sheet_number - 1)
224  sentences_info, unknown_words = read_excel_and_find_words(sentences_filename, sentences_sheet_number, article_data_filename, article_da
225  updated_unknown_words = search_words_and_print_info(sentences_filename, words_sheet_number, unknown_words)
226
227  output_filename = 'results.txt'
228  with open(output_filename, 'w', encoding='utf-8') as file:
229   file.write('Υποκείμενο --> 1 \n')
230   file.write('Ρήμα --> P \n')
231   file.write('Άμεσο αντικείμενο --> 2 \n')
232   file.write('Έμμεσο αντικείμενο --> 3 \n')
233   file.write('='*50 + '\n')
234
235   # Εκτύπωση των αποτελεσμάτων
236   for idx, sentence_info in enumerate(sentences_info):
237      print(f"Πρόταση {idx + 1}    : {sentence_info['sentence']}")
238      file.write(f"Πρόταση {idx + 1}: {sentence_info['sentence']}\n")
239      l=0
240      k=0
241      g=0
242      h=0
243      j=0
244      p=0
245      o=0
246      i=0
```

Αρχικά ορίζονται οι μεταβλητές που περιέχονται τα ονόματα των αρχείων Excel, με τους αριθμούς των φύλλων που υπάρχουν οι προτάσεις, τα άρθρα, τα ρήματα, τις αντωνυμίες και άλλες λέξεις. Παρακάτω καλούνται οι συναρτήσεις όπου αναλύθηκαν λεπτομερώς παραπάνω μαζί με τίς λοιπές πληροφορίες που χρειάζονται για την επεξεργασία της εκάστοτε συνάρτησης.

Μετά την επεξεργασία των δεδομένων ανοίγει ένα αρχείο κειμένου (“results.txt”) στο οποίο θα περνάνε τα αποτελέσματα των προτάσεων. Το αρχείο επεξηγεί τι αντιπροσωπεύει κάθε αριθμός ή γράμμα. Το “1” για το υποκείμενο, το “P” για το ρήμα, το “2” για το άμεσο αντικείμενο, και το “3” για το έμμεσο αντικείμενο. Επιπλέον γράφεται μια γραμμή με το σύμβολο “=” για να ξεχωρίζει η εισαγωγή από τα αποτελέσματα.

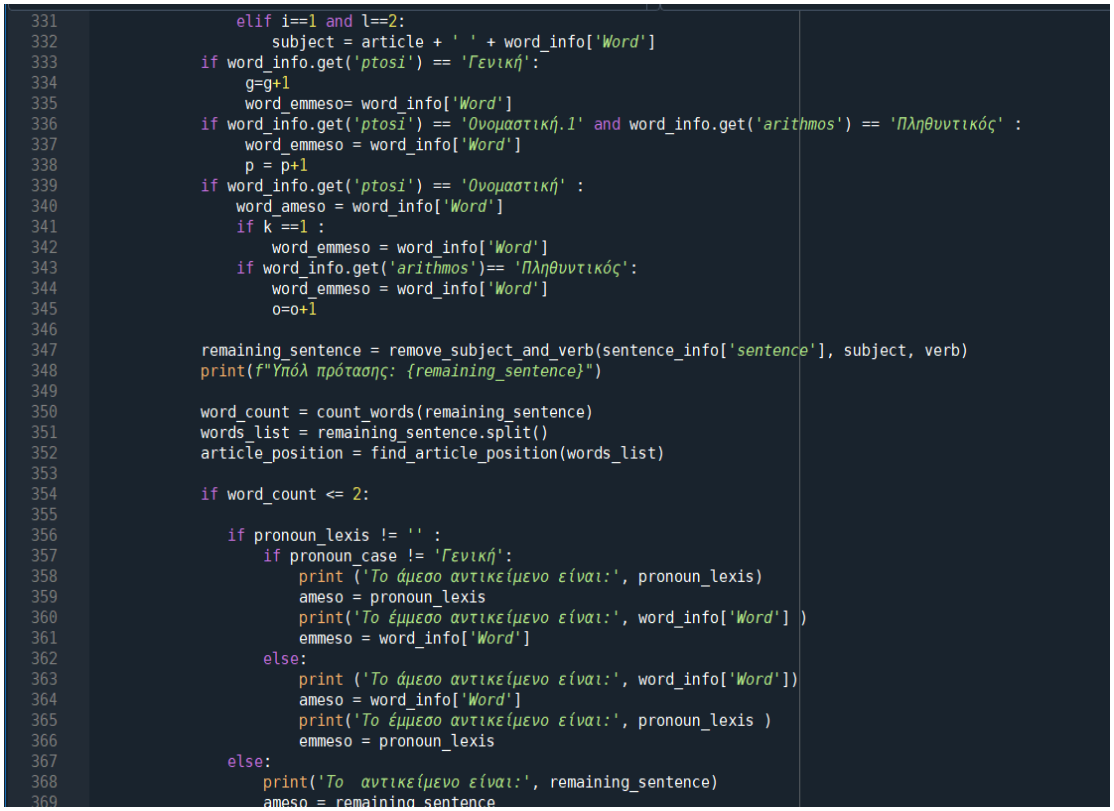
Στην συνέχεια σε έναν βρόχο “ for” τρέχει την λίστα “ sentences_info” ενώ για κάθε πρόταση εκτυπώνει και καταγράφει στο αρχείο κειμένου την πρόταση και τον αριθμό της. Επίσης αρχικοποιούνται κάποιες μεταβλητές οι οποίες θα χρειαστούν στην συνέχεια του προγράμματος.

```
    word_ameso = ''
    word_emmeso = ''
    verb = ""
    ameso_antikeimeno = ""
    emmeso_antikeimeno = ""
    pronoun_lexis = ''
    pronoun_case=''
    article_emmeso = ''
    subject =''
    article_ameso=''
    ameso = '  -  '
    emmeso= '  -  '

    for word_info in sentence_info['word_info_list']:

        print(f"Λέξη         : {word_info['Word']}")
        print(f"Τύπος        : {word_info['Type']}")
        if word_info['Type']=='Pronoun':
            pronoun_lexis = word_info['Word']
            pronoun_case = word_info['Details'][1]
        if word_info['Type'] == 'Article':
            case = word_info['Details'].get('Case', None)
            if case == 'ονομαστική':
                i=i+1
                article = word_info['Word']

                article_gender = word_info['Details'].get('Gender', None)
            if case == 'αιτιατική':
                k=k+1

                if k==1:
                    article_emmeso = word_info['Word']

                if k==2:

                    article_ameso = word_info['Word']
            if case == 'ονομαστική/αιτιατική' :
                j=j+1

                article = word_info['Word']
                article_ameso = word_info['Word']
```

Αυτό το κομμάτι του κώδικα θέτει είτε το κενό σύμβολο είτε την “-” για να αρχικοποιήσει τις μεταβλητές που επεξεργάζονται στην συνέχεια του προγράμματος. Οι μεταβλητές αυτές είναι το ρήμα, το υποκείμενο, το άμεσο και το έμμεσο αντικείμενο, καθώς και τα άρθρα και τις αντωνυμίες. Στη συνέχεια ο κώδικας τρέχει την λίστα με τις λέξεις των προτάσεων και για κάθε λέξη εκτυπώνει τον τύπο της, δηλαδή αν είναι άρθρο, αντωνυμία κτλ. Αν η λέξη είναι αντωνυμία, τότε αποθηκεύει την λέξη και την πτώση της. Αν η λέξη είναι άρθρο τότε η πτώση της καθορίζει αν είναι υποκείμενο, άμεσο ή έμμεσο αντικείμενο. Οι μεταβλητές αυτές μπορούν να χρησιμοποιηθούν για την ανάλυση της πρότασης παρακάτω.

```
            if case == 'γενική':
                h = h + 1
                article_emmeso = word_info['Word']
            if case == 'αιτιατική' and p==0 :
                article_emmeso = word_info['Word']

        if word_info['Type'] == 'Verb':
            i=-1 #  Flag για να πεταει εξω τις εσφαλμενες τιμες.
            verb = word_info['Word']
            print('Το ρήμα ειναι:', verb)

        if word_info['Type'] == 'Unknown':
            print(f"Μέρος        : {word_info.get('meros', 'Δεδομένα δεν βρέθηκαν')}")
            print(f"Πτώση        : {word_info.get('ptosi', 'Δεδομένα δεν βρέθηκαν')}")
            print(f"Γένος        : {word_info.get('genos', 'Δεδομένα δεν βρέθηκαν')}")

            if word_info.get('ptosi') == 'Ονομαστική' :
                l = l + 1

                if l==2:
                   word_ameso = word_info['Word']
                if l==3:
                    word_emmeso = word_info['Word']

            if word_info.get('ptosi') == 'Ονομαστική':
                i=i+1

                if h==1:
                    word_ameso = word_info['Word']

                if word_info.get('genos') != article_gender:
                    i=0

                if i == 2 :

                    subject = article + ' ' + word_info['Word']
                    previous_word = word_info['Word']

                elif i == 3 :
                    subject = article + ' ' + previous_word + ' ' + word_info['Word']
                    i=0
```

Αρχικά εξετάζει αν η πτώση ενός άρθρου είναι στην γενική, αν είναι αυξάνει έναν μετρητή και το αποθηκεύει ως έμμεσο αντικείμενο. Στην περίπτωση που η πτώση του είναι στην αιτιατική και δεν έχει χρησιμοποιηθεί ακόμα (P=0), το άρθρο αποθηκεύεται πάλι ως έμμεσο. Ύστερα αν η λέξη είναι ρήμα τότε αποθηκεύεται στην μεταβλητή για το ρήμα και ο μετρητής γίνεται -1 για να αποφύγει τις εσφαλμένες τιμές. Στις άγνωστες λέξεις εκτυπώνει τις πληροφορίες με την πτώση, μέρος του λόγου και γένος. Αν η πτώση είναι ονομαστική, αυξάνεται ένας μετρητής και ανάλογα με τη σειρά της λέξης αποθηκεύεται είτε ως άμεσο είτε ως έμμεσο αντικείμενο. Επίσης αν το γένος της λέξης δεν ταιριάζει με το γένος του άρθρου, ο μετρητής μηδενίζεται. Όταν ο μετρητής φτάσει σε συγκεκριμένες τιμές, ο κώδικας δημιουργεί το υποκείμενο της πρότασης, που μπορεί να περιέχει ένα άρθρο με τις αντίστοιχες λέξεις. Αυτή η δομή του κώδικα καθορίζει το υποκείμενο με βάση την προηγούμενη λέξη. Αν το άρθρο είναι στην ονομαστική και η αντίστοιχη λέξη στην ονομαστική τότε είναι υποκείμενο, ειδάλλως αντικείμενο.



```
                elif i==1 and l==2:
                    subject = article + ' ' + word_info['Word']
            if word_info.get('ptosi') == 'Γενική':
                 g=g+1
                 word_emmeso= word_info['Word']
            if word_info.get('ptosi') == 'Ονομαστική.1' and word_info.get('arithmos') == 'Πληθυντικός' :
                 word_emmeso = word_info['Word']
                 p = p+1
            if word_info.get('ptosi') == 'Ονομαστική' :
                word_ameso = word_info['Word']
                if k ==1 :
                    word_emmeso = word_info['Word']
                if word_info.get('arithmos')== 'Πληθυντικός':
                    word_emmeso = word_info['Word']
                    o=o+1

            remaining_sentence = remove_subject_and_verb(sentence_info['sentence'], subject, verb)
            print(f"Υπόλ πρότασης: {remaining_sentence}")

            word_count = count_words(remaining_sentence)
            words_list = remaining_sentence.split()
            article_position = find_article_position(words_list)

            if word_count <= 2:

               if pronoun_lexis != '' :
                    if pronoun_case != 'Γενική':
                        print ('Το άμεσο αντικείμενο είναι:', pronoun_lexis)
                        ameso = pronoun_lexis
                        print('Το έμμεσο αντικείμενο είναι:', word_info['Word'] )
                        emmeso = word_info['Word']
                    else:
                        print ('Το άμεσο αντικείμενο είναι:', word_info['Word'])
                        ameso = word_info['Word']
                        print('Το έμμεσο αντικείμενο είναι:', pronoun_lexis )
                        emmeso = pronoun_lexis
               else:
                    print('Το  αντικείμενο είναι:', remaining_sentence)
                    ameso = remaining_sentence
```

Όπως και πριν, ο κώδικας συνεχίζει την ανάλυση της πρότασης με στόχο να εντοπίσει και να αποθηκεύσει σε μια μεταβλητή τα έμμεσα και άμεσα αντικείμενα. Εάν η πτώση είναι γενική, η λέξη αποθηκεύεται ως έμμεσο αντικείμενο. Αν η πτώση είναι ονομαστική και ο αριθμός είναι πληθυντικός, η λέξη αποθηκεύεται ως έμμεσο αντικείμενο και αυξάνεται ο μετρητής “p”. Αν τώρα η πτώση είναι ονομαστική, η λέξη αποθηκεύεται ως άμεσο αντικείμενο και, αν είναι το άρθρο σε αιτιατική πτώση, η λέξη αυτή χρησιμοποιείται ως έμμεσο αντικείμενο. Στην συνέχεια η πρόταση αφαιρεί το υποκείμενο και το ρήμα και αυτό που μένει αποθηκεύεται στην μεταβλητή remaining_sentence. Αν ο αριθμός των λέξεων της μεταβλητής είναι μικρότερος ή ίσος με 2, γίνεται έλεγχος για αντωνυμίες ή αντικείμενα. Ανάλογα τα άμεσα και έμμεσα αντικείμενα βρίσκονται και εκτυπώνονται. Αν δεν υπάρχουν αντωνυμίες τότε η υπόλοιπη πρόταση είναι αντικείμενο.

```
370             else:
371                 if word_count==3:
372 
373                     if k==1 and l == 2 and o==0 and article_position == 1 :
374                         emmeso_antikeimeno = article_emmeso  + ' ' + word_emmeso
375                         print('Το έμμεσο αντικείμενο είναι:', emmeso_antikeimeno)
376                         emmeso = emmeso_antikeimeno
377                         remaining_sentence = remaining_sentence.replace(emmeso_antikeimeno, '')
378 
379                         print('Το άμεσο αντικείμενο είναι:', remaining_sentence)
380                         ameso = remaining_sentence
381                         break
382                     if k==1 and l == 2 and o==0 and article_position == 0:
383                         ameso_antikeimeno = article_emmeso  + ' ' + word_emmeso
384                         print('Το άμεσο αντικείμενο είναι:', ameso_antikeimeno)
385                         ameso = ameso_antikeimeno
386                         remaining_sentence = remaining_sentence.replace(ameso_antikeimeno, '')
387                         print('Το έμμεσο αντικείμενο είναι:', remaining_sentence)
388                         emmeso = remaining_sentence
389                         break
390                     if article_position == 0 and k==1 :
391                         emmeso_antikeimeno = article_emmeso + ' ' + word_emmeso
392                         print('Το έμμεσο αντικείμενο είναι:', emmeso_antikeimeno)
393                         emmeso = emmeso_antikeimeno
394                         remaining_sentence = remaining_sentence.replace(emmeso_antikeimeno, '')
395                         print('Το άμεσο αντικείμενο είναι:', remaining_sentence)
396                         ameso = remaining_sentence
397                         break
398                     elif p==1 and k==1 and l!=2:
399                         emmeso_antikeimeno = article_emmeso + ' ' + word_emmeso
400                         print('Το έμμεσο αντικείμενο είναι:',emmeso_antikeimeno)
401                         emmeso= emmeso_antikeimeno
402                         ameso_antikeimeno = word_ameso
403                         print('Το άμεσο αντικείμενο είναι:', ameso_antikeimeno)
404                         ameso = ameso_antikeimeno
405                         break
406                     else:
407                         print('Το αντικείμενο είναι:', remaining_sentence)
408                         ameso= remaining_sentence
```

Αυτό το κομμάτι του κώδικα αναλύει την περίπτωση όπου η υπόλοιπη πρόταση είναι τρείς λέξεις. Με βάση τους διάφορους συνδυασμούς συνθηκών εκτυπώνεται αν είναι άμεσο ή έμμεσο αντικείμενο. Οι πολλοί μετρητές χρησιμοποιούνται για την αντιμετώπιση των άγνωστων λέξεων (unknown_words). Για παράδειγμα η υπόλοιπη πρόταση που έμεινε ήταν *Το μαύρο ποντίκι*, οι άγνωστες λέξεις του Excel είναι "μαύρο ποντίκι". Εφόσον το άρθρο είναι σε αιτιατική πτώση, η επόμενη λέξη (*μαύρο*) δεν αναγνωρίζεται από το πρόγραμμα τι πτώση είναι. Αυτό συμβαίνει γιατί και στην ονομαστική και στην αιτιατική είναι η ίδια λέξη. Έτσι με βάση την προηγούμενη λέξη εμφανίζει την σωστή πτώση με την βοήθεια των μετρητών. Επομένως οι διαφορετικές συνθήκες εξυπηρετούν το σφάλμα της διαφορετικής πτώσεως και γενικότερα τα σφάλματα των unknown_words όπου αντιμετωπίζονται στην relational grammar.

```
                if word_count>=4:
                    if g == 1 and h==1 and j==1 :
                        emmeso_antikeimeno = article_emmeso + ' '+ word_emmeso
                        print('Το έμμεσο αντικείμενο είναι:',emmeso_antikeimeno)
                        emmeso = emmeso_antikeimeno
                        ameso_antikeimeno = article_ameso + ' ' + word_info['Word']
                        print('Το άμεσο αντικείμενο είναι:', ameso_antikeimeno)
                        ameso = ameso_antikeimeno
                        break
                    if k==1 and l==3:
                        emmeso_antikeimeno = article_emmeso + ' ' + word_emmeso
                        print('Το έμμεσο αντικείμενο είναι:',emmeso_antikeimeno)
                        emmeso= emmeso_antikeimeno
                        remaining_sentence = remaining_sentence.replace(emmeso_antikeimeno, '')
                        print('Το άμεσο αντικείμενο είναι:', remaining_sentence)
                        ameso = remaining_sentence
                    if k==2:
                        ameso_antikeimeno = article_ameso +' '+ word_ameso
                        print('Το άμεσο αντικείμενο είναι:', ameso_antikeimeno)
                        ameso = ameso_antikeimeno
                        remaining_sentence = remaining_sentence.replace(ameso_antikeimeno, '')
                        print('Το έμμεσο αντικείμενο είναι:',remaining_sentence)
                        emmeso = remaining_sentence
    print('Το υποκείμενο είναι:', subject)
    print('Το ρήμα ειναι:', verb)
    file.write(f' 1 : {subject} \n')
    file.write(f' P : { verb}\n')
    file.write(f' 2 : {ameso}\n')
    file.write(f' 3 : {emmeso}\n')
    file.write('='*50 + '\n')
    print('-' * 50)

print("Τα αποτελέσματα αποθηκεύτηκαν στο results.txt")
print('Τέλος προγράμματος.')
```

Σε αυτό το τμήμα του κώδικα όπως και πριν χρησιμοποιούνται πάλι οι μετρητές, όμως σε αυτή την περίπτωση χρησιμοποιούνται για remaining_sentence μεγαλύτερο ή ίσο του 4. Ανάλογα με τους μετρητές παράγεται το επιθυμητό αποτέλεσμα, το οποίο στην προκείμενη περίπτωση μπορεί να είναι άμεσο ή έμμεσο αντικείμενο. Επιπλέον το υποκείμενο, το ρήμα, τα αντικείμενα εκτυπώνονται και καταγράφονται σε ένα αρχείο κειμένου “results.txt”. Τέλος ο κώδικας ολοκληρώνεται όταν επιβεβαιώνεται ότι τα αποτελέσματα έχουν αποθηκευτεί.

```
Υποκείμενο --> 1
Ρήμα --> P
Άμεσο αντικείμενο --> 2
Έμμεσο αντικείμενο --> 3
================================================
Πρόταση 1: Η γάτα έφαγε το ποντίκι
 1 : Η γάτα
 P : έφαγε
 2 : το ποντίκι
 3 :   -
================================================
Πρόταση 2: Η άσπρη γάτα έφαγε το μαύρο ποντίκι
 1 : Η άσπρη γάτα
 P : έφαγε
 2 : το μαύρο ποντίκι
 3 :   -
================================================
Πρόταση 3: Το ποντίκι έφαγε η γάτα
 1 : η γάτα
 P : έφαγε
 2 : Το ποντίκι
 3 :   -
================================================
Πρόταση 4: Ο Νίκος έδωσε ένα λουλούδι  στην Μαρία
 1 : Ο Νίκος
 P : έδωσε
 2 : ένα λουλούδι
 3 : στην Μαρία
```

# 5. ΑΞΙΟΛΟΓΗΣΗ ΛΕΙΤΟΥΡΓΙΑΣ

Για την υλοποίηση της εργασίας σε Python, έγινε μια προσέγγιση μεθοδική η οποία είναι βασισμένη σε έλεγχο και δοκιμές. Για την εξοικονόμηση χρόνου δημιουργήθηκαν πολλά μικρά τμήματα κώδικα και πολλαπλές συναρτήσεις για τον ευκολότερο έλεγχο και κατά συνέπεια την ευκολότερη λύση του προβλήματος. Αν τα μικρά τμήματα εκτελούσαν σωστά την διαδικασία που έπρεπε συνενώνονταν σταδιακά ένα-ένα με αποτέλεσμα να συμπληρώνεται με γοργούς ρυθμούς από ότι αν γινόταν όλο μονομιάς. Αυτή η διαδικασία επέτρεψε να ελέγχει το κάθε τμήμα ξεχωριστά, εξασφαλίζοντας ότι το κάθε κομμάτι λειτουργεί ορθά. Αυτός ο τρόπος προσέγγισης οργάνωσε τις συναρτήσεις σε μικρά τμήματα με συγκεκριμένο ρόλο και μπόρεσε να ελέγχει αυτόνομα τα τυχόν σφάλματα ή αναγκαίες προσαρμογές. Μετά την ολοκλήρωση του κάθε κομματιού γινόντουσαν οι απαραίτητες δοκιμές στο Anaconda Kernel, που βρίσκεται μέσα στην εφαρμογή της Spyder και με αρκετά print γινόταν αποσφαλμάτωση. Μόλις εμφάνιζε σωστά όλα τα μηνύματα μέσα στο kernel, ενώνονταν όλα τα κομμάτια τμήμα-τμήμα, με αποτέλεσμα να παραχθεί το τελικό μέρος.

Οι μελλοντικές βελτιώσεις που μπορούν να γίνουν θα ήταν η μείωση του αριθμού των γραμμών ή η αύξηση της πολυπλοκότητας του κώδικα. Για να επιτευχθεί αυτό θα πρέπει να γίνει χρήση σύνθετων συναρτήσεων, όπου εκτελούν πιο πολλές λειτουργίες από μια. Μερικές εντολές είναι οι Map(), Filter() και reduce(), οι οποίες μπορούν να αντικαταστήσουν βρόχους επανάληψης και κατά συνέπεια να μειωθεί ο κώδικας. Τέτοιες τεχνικές και πολλές άλλες θα κάνει τον κώδικα πιο αποδοτικό.

Μια άλλη προσέγγιση θα μπορούσε να είναι η αξιοποίηση βιβλιοθηκών, οι οποίες έχουν φτιαχτεί για να κάνει πιο γρήγορο τον κώδικα. Για παράδειγμα η βιβλιοθήκη numpy επεξεργάζεται μεγάλες ποσότητες δεδομένων σε λιγοστές γραμμές κώδικα. Αυτοί οι τρόποι θα προσφέρουν μεγαλύτερη αποδοτικότητα στο μέλλον.

# 6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η υλοποίηση του κώδικα και της βάσης δεδομένων για την επεξεργασία και την ανάλυση των γλωσσικών δεδομένων πραγματοποιήθηκε με επιτυχία. Αρχικά στη βάση δεδομένων καταχωρήθηκε το λεξικό του Τριανταφυλλίδη και αναμορφώθηκε για να διευκολύνει την αναζήτηση και να εμφανίζει τις πληροφορίες στο κύριο πρόγραμμα. Οι πληροφορίες οργανώθηκαν με τέτοιο τρόπο έτσι ώστε να είναι ανά κατηγορία της λέξης (άρθρα, ουσιαστικά, επίθετα, αντωνυμίες και ρήματα) και εμπλουτίστηκαν με πτώση, γένος, φωνή και χρόνο για να υποστηρίξουν την σωστή αναγνώριση των λέξεων.

Στην συνέχεια αναπτύχθηκε κώδικας με την βοήθεια βιβλιοθηκών όπως οι Pandas, Re, Unicodedata. Αυτές οι βιβλιοθήκες διευκόλυναν την επεξεργασία των δεδομένων και κατ' επέκταση την αναγνώριση των γλωσσικών στοιχείων. Έπειτα οι συναρτήσεις που δημιουργήθηκαν μέσα στο πρόγραμμα βελτίωσαν την αποδοτικότητα στην αναζήτηση, επαλήθευση και μετατροπή των λέξεων με βάση τους γλωσσικούς κανόνες.

Μερικές βιβλιοθήκες όπως η NLTK και spaCy δεν λειτούργησαν στον κώδικα λόγω κάποιων προβλημάτων που προέκυψαν. Παρότι η εργασία υλοποιήθηκε κανονικά, αυτές οι βιβλιοθήκες θα διευκόλυναν ακόμη πιο γρήγορα τον κώδικα. Οι βιβλιοθήκες δεν λειτούργησαν διότι μπορεί να υπήρχαν προβλήματα συμβατότητας ή έλλειψη στην ρύθμιση του περιβάλλοντος. Συχνά τέτοιου είδους βιβλιοθήκες μπορεί ακόμη να χρειάζονται συγκεκριμένες εκδόσεις Python ή μπορεί να χρειαζόντουσαν πρόσθετα αρχεία για να κατέβουν. Έτσι αν δεν γίνουν οι κατάλληλες προεργασίες δεν μπορούν να δουλέψουν.

Συνολικά χρειάστηκαν περίπου 6 μήνες για την περάτωση της εργασίας με καθημερινή απασχόληση περίπου 4 ωρών. Αυτό αντιστοιχεί σε περίπου 3 μήνες εργασίας και συνολικά σε 720 ώρες. Με την διαχείριση αυτή προσφέρθηκε σημαντική συνεισφορά στις δεξιότητές μου για την ολοκλήρωση της εργασίας.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


1. Beazley D. *Python Essential Reference*. 4th ed. Addison-Wesley Professional; 2009. Available from: https://books.google.gr/books?hl=el&lr=&id=Chr1NDlUcI8C&oi=fnd&pg=PR7&dq=Beazley+D.+Python+Essential+Reference.&ots=OFCBtji_Fp&sig=jr7M-tpoz8QuTxb_9JWT4o5Kdw4&redir_esc=y#v=onepage&q&f=false
2. Bird S, Klein E, Loper E. *Natural Language Processing with Python*. O'Reilly Media; 2009. Available from: https://www.nltk.org/book/
3. Big Blue Academy. Υπολογιστική όραση. Available from: https://bigblue.academy/gr/upologistiki-orasi
4. C++ Reference. Available from: https://en.cppreference.com/w/
5. Google. *Google Sheets*. Available from: https://izipen.gr/blog/ti-einai-to-google-sheets/
6. Honnibal M, Montani I. *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*; 2017. Available from: https://spacy.io/usage/spacy-101
7. Joshi P. *Machine Learning with Java* [Internet]. Birmingham, UK: Packt Publishing; 2017 [cited 2024 Sep 13]. Available from: https://books.google.gr/books?hl=el&lr=&id=WPFsDwAAQBAJ&oi=fnd&pg=PP1&dq=Joshi+P.+Machine+Learning+with+Java.&ots=Hm3v015VRG&sig=-JOPYJBIyXo6Ooyt2awQGUkO4CY&redir_esc=y#v=onepage&q=Joshi%20P.%20Machine%20Learning%20with%20Java.&f=false
8. Γεωργούλη Κ. *Τεχνητή Νοημοσύνη*. ΣΕΑΒ, 2015. Kallipos. *Relational Grammar*. Available from: http://repfiles.kallipos.gr/html_books/93/07a-main.html
9. Καρολίδης Δ., Ξαρχάκος Κ. *Microsoft Excel 2016: θεωρία, συναρτήσεις, προγραμματισμός με VBA εφαρμογές*. 1η έκδ. Κλειδάριθμος; 2016.
10. Microsoft. *Βασικές πληροφορίες για τις βάσεις δεδομένων*. Available from: https://support.microsoft.com/el-gr/topic/%CE%B2%CE%B1%CF%83%CE%B9%CE%BA%CE%AD%CF%82-%CF%80%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%AF%CE%B5%CF%82-%CE%B3%CE%B9%CE%B1-%CF%84%CE%B9%CF%82-%CE%B2%CE%AC%CF%83%CE%B5%CE%B9%CF%82-%CE%B4%CE%B5%CE%B4%CE%BF%CE%BC%CE%AD%CE%BD%CF%89%CE%BD-a849ac16-07c7-4a31-9948-3c8c94a7c204
11. SAP. *What is machine learning?*. Available from: https://www.sap.com/greece/products/artificial-intelligence/what-is-machine-learning.html
12. spaCy. *spaCy 101: Usage*. Available from: https://spacy.io/usage/spacy-101
13. Mindstorm. *Τι είναι η Access*. Available from: https://mindstorm.gr/ti-einai-h-access/

14. Hands-on NLTK Tutorial with the Greek Script. Available from: https://github.com/hb20007/hands-on-nltk-tutorial/blob/main/7-1-NLTK-with-the-Greek-Script.ipynb
15. OpenDataScience. *The 30 Most Useful Python Libraries for Data Engineering* [Internet]. Available from: https://opendatascience.com/the-30-most-useful-python-libraries-for-data-engineering/
16. iGuru.gr. *Τι είναι και πώς λειτουργεί η βαθιά μάθηση*. Available from: https://iguru.gr/einai-kai-pos-leitourgei-vathia-mathisi/
17. Blake, Barry J. *Relational Grammar*. London: Routledge, 1990.