Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

## Προαιρετική Εργασία σε Python



**Φοιτητής:** Ανδρέας Κάλλιστρος **ΑΜ:** 1054351



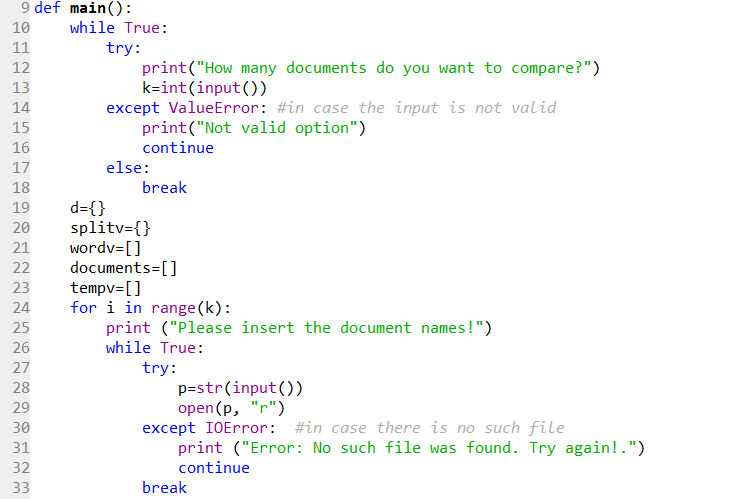
**Αναφορά στην Προαιρετική Εργαστηριακή Άσκηση Εξαμήνου (2019)**

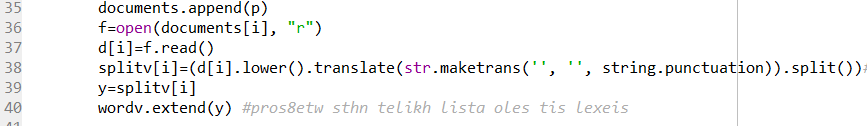
# Εισαγωγή

Για το project αυτό κλήθηκα να υλοποιήσω ένα πρόγραμμα σε Python που να δέχεται ως είσοδο N αρχεία κειμένου και να υπολογίζει τα ποσοστά ομοιότητας των κειμένων, μέσω της ομοιότητας συνημίτονου, και να εμφανίζει τα Κ ζευγάρια με την μεγαλύτερη ομοιότητα

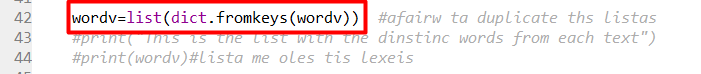
# Περιγραφή κώδικα

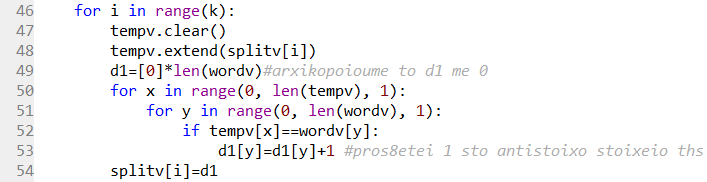
Αρχικά ζητείται να δοθεί ο αριθμός των κειμένων που θα συγκριθούν, τα ονόματα των αρχείων και αρχικοποιούνται διάφορες λίστες και dictionaries που θα χρησιμοποιηθούν. Επίσης με διάφορους ελέγχους βεβαιωνόμαστε ότι οι είσοδοι έχουν την σωστή μορφή. Στην λίστα **documents[]** αποθηκεύονται τα ονόματα των αρχείων που δόθηκαν.



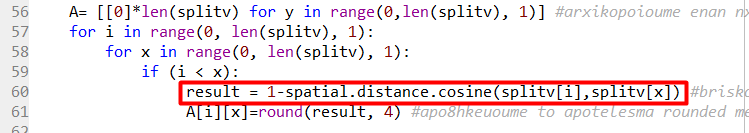
Στο dictionary **d{}** αποθηκεύονται τα κείμενα των αρχείων και στην συνέχεια μέσω της συνάρτησης **split()** χωρίζουμε το κάθε κείμενο σε λέξεις οι οποίες αποθηκεύονται στο dictionary **splitv{}**. Επίσης όλοι οι κεφαλαίοι χαρακτήρες γίνονται μικροί και αφαιρούνται όλα τα σημεία στίξης (με την βοήθεια συνάρτησης της βιβλιοθήκης **string**). Τέλος με την βοήθεια της συνάρτησης **extend()** εισάγουμε όλες τις λέξεις στην λίστα **wordv[].**

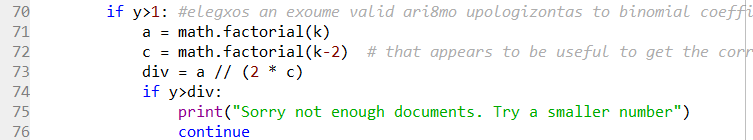
Αφού γίνετε αυτό για όλα τα κείμενα εισόδου έχουμε στη λίστα **wordv[]** όλες τις διακριτές λέξεις από όλα τα κείμενα. Όμως πρέπει να αφαιρέσουμε τις διπλότυπες λέξεις. Για να το πετύχουμε αυτό εκμεταλλευόμαστε την ιδιότητα που έχουν τα dictionaries στην python η οποία δεν επιτρέπει να υπάρχουν keys τα οποία να έχουν την ίδια τιμή. Έτσι μετατρέπουμε την λίστα μας σε dictionary με κλειδιά τα στοιχεία της λίστας wordv[], αφαιρώντας έτσι τα διπλότυπα, και ξαναμετατρέπουμε το καινούριο αυτό dictionary πάλι σε λίστα.

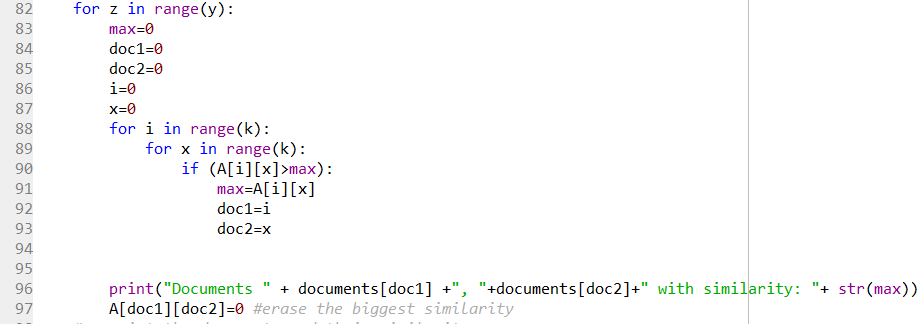


Στο dictionary splitv{} υπάρχουν τα κείμενα χωρισμένα σε λέξεις. Τώρα θα δημιουργήσουμε έναν διάνυσμα d1 με μήκος όσο και το μήκος του wordv[] και κάθε φορά που συναντάμε μια λέξη σε ένα κείμενο θα προσθέτουμε 1 στο αντίστοιχο κελί του d1. Αφού διαπεράσουμε όλες τις λέξεις ενός κειμένου θα αποθηκεύσουμε το d1 στην κατάλληλη θέση στο splitv{} και αφού αδειάσουμε το d1 θα επαναλάβουμε την διαδικασία που περιγράψαμε για κάθε κείμενο. 

Στο splitv{} θα είναι αποθηκευμένα τα διανύματα του κάθε κειμένου. Στην συνέχεια δημιουργούμε ένα array με διαστάσεις nxn όπου n είναι το πλήθος των κείμενων. Εκεί αποθηκεύουμε το cosine similarity έτσι ώτσε για παράδειγμα στο A[0][1] να είναι αποθηκευμένο το cosine similarity του αρχείου 0 και 1, και ούτω καθεξής. Τα κελιά κάτω από την κύρια διαγώνιο θα είναι 0 (πχ Α[1][0]). Για να υπολογίσουμε το cosine similarity χρησιμοποιούμε την βιβλιοθήκη splatial (την οποία κάνουμε import στην αρχή).

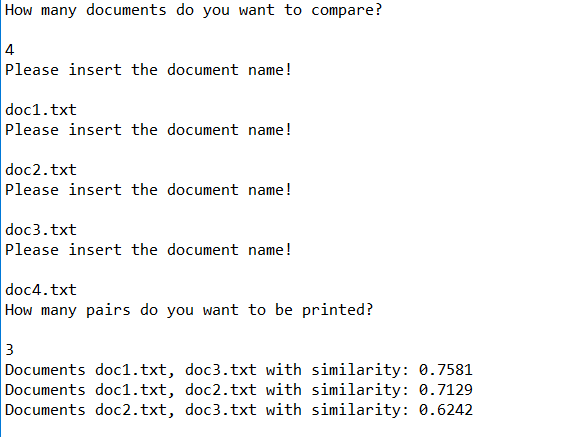


Τέλος ζητάμε από τον χρήστη το Κ το οποίο θα πρέπει να είναι integer και μικρότερο από το binomial coefficient (). Για να υπολογίσουμε το N ανά 2 χρησιμοποιήσαμε την συνάρτηση factorial της βιβλιοθήκης math. 

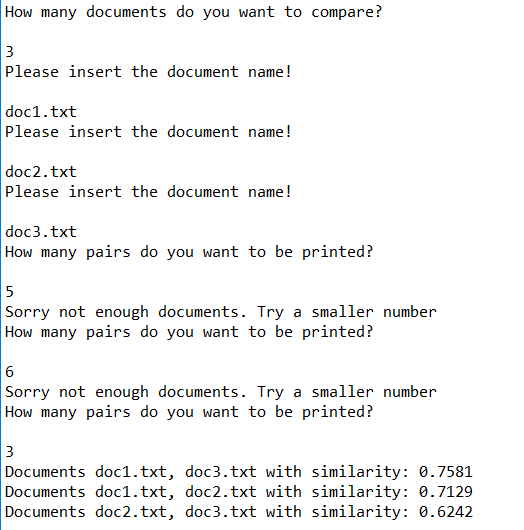
Για να τυπώσουμε τα ζευγάρια με την μεγαλύτερη ομοιότητα διατρέχουμε τον Α και τυπώνουμε τα Κ με την μεγαλύτερη ομοιότητα 

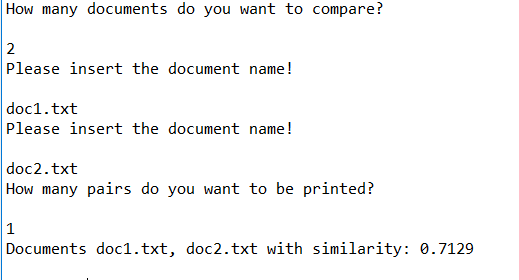
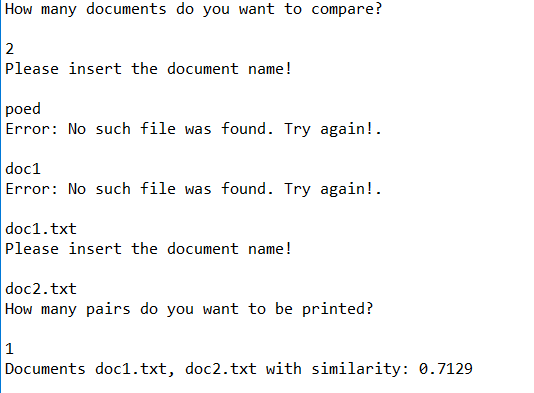
# Παραδείγματα εκτέλεσης

* Εκτέλεση του προγράμματος με είσοδο 4 κείμενα σχετικά με την XML από το Wikipedia.



* Εκτέλεση με 3 αρχεία που να ζητάει Κ το οποίο δεν είναι έγκυρο



* Εκτέλεση με τα δύο αρχεία της εκφώνησης 
* Μη έγκυρο όνομα αρχείου 
* Διάφορες άλλες περιπτώσεις λαθών 