**Main.py:** Αρχικοποιεί τον counter για να μετρήσει το πλήθος των σελίδων που κάνει crawl, το links που είναι τα links που βρίσκει ο crawler από κάθε σελίδα και ένα dictionary με όνομα data που κρατάει προσωρινά τα δεδομένα.

Το data έχει ως key το link και ως value το text από το συγκεκριμένο url. Τα δεδομένα αυτά περνάνε στον indexer.

**MyCrawler.py** :

Χρησιμοποιεί 2 σύνολα, το queue και το crawled για να αποθηκεύει τα link που έχει να κάνει crawl και αυτά που έχει ήδη κάνει.

Χρησιμοποιούμε set για να μην έχουμε διπλότυπα.

def \_\_init\_\_(self, url, num\_of\_pages, delete\_data, num\_of\_threads,  
 data\_directory="Data", queue\_file="/queue.txt", crawled\_file="/crawled.txt")

Κατασκευαστής του crawler.

def scrape\_links(url):

Παίρνει σαν όρισμα ένα url, κάνει το request με την βιβλιοθήκη requests και παίρνει όλα τα link από αυτό χρησιμοποιώντας την beautifulSoup4. Μετά παίρνει τα κείμενα από όλα τα p tags που βρίσκονται στο συγκεκριμένο url και τα καθαρίζει με ένα regular expression.

def boot(self):

Αρχικοποιεί τους φακέλους που χρειάζεται το πρόγραμμα για να τρέξει. Σε περίπτωση που το όρισμα delete\_data είναι 1, αναδημιουργεί τους φακέλους και τα αρχεία σβήνοντας τα περιεχόμενά τους.

Σε διαφορετική περίπτωση, φορτώνει τα σύνολα queue και crawled και συνεχίζει.

def print\_crawler(self):

Τυπώνει το στιγμιότυπο του crawler

def queue\_to\_set(self):

Χρησιμοποιείται όταν το όρισμα delete\_data είναι 0, για να γεμίσει το set queue από το αντίστοιχο αρχείο.

def crawled\_to\_set(self):

Χρησιμοποιείται όταν το όρισμα delete\_data είναι 0, για να γεμίσει το set crawled από το αντίστοιχο αρχείο.

def queue\_to\_file(path, data):

Χρησιμοποιείται για να αποθηκεύσει τα links που έχει να κάνει crawl σε αρχείο, μόλις σταματήσει η λειτουργεία του crawler. (Αν χρησιμοποιηθούν threads ελέγχει με κλειδαριές(lock\_queue) αν το αρχείο γράφεται από κάποιο άλλο thread για έλεγχο ταυτοχρονισμού).

def crawled\_to\_file(path, data):

Χρησιμοποιείται για να αποθηκεύσει τα links που έχει κάνει crawl σε αρχείο, μόλις σταματήσει η λειτουργεία του crawler. (Αν χρησιμοποιηθούν threads ελέγχει αν το αρχείο γράφεται από κάποιο άλλο thread με τη χρήση της κλειδαριάς lock\_crawled).

**Functions.py:**

def create\_directory(directory\_name="Data"):

Δημιουργεί ένα φάκελο με όνομα directory\_name. Προεπιλεγμένο όνομα Data.

def create\_data\_files(STARTING\_URL="https://www.google.gr"):

Δημιουργεί τα αρχεία queue και crawled και προσθέτει το STARTING\_URL στο αρχείο queue

def write\_file(path, data):

Γράφει τα δεδομένα data στο αρχείο που ορίζεται από την διαδρομή path.

def append\_to\_file(path, data):

Κάνει append στο αρχείο που βρίσκεται στην διαδρομή Path, τα δεδομένα data.

def delete\_data(path):

Διαγράφει τα δεδομένα από το αρχείο που βρίσκεται στη διαδρομή path.

**INDEXER**: O indexer δέχεται κάθε φορα από τον **Crawler** ένα url και ένα κείμενο που αντιστοιχεί σε εκείνο το url και αφού πρώτα το καθαρίσει (αφαιρέσει σημεία στιξης κλπ. με την βοήθεια της βιβλιοθήκης re) και κάνει όλα τα κεφαλαια γράμματα, μικρά, το κάνει tokenize. Στην συνέχεια δημιουργεί τον ανεστραμένο κατάλογο (με tf όμως). Για την δημιουργία ανεστραμένου καταλόγου χρησημοποιείται ένα temp\_dict, το οποίο ανανεώνεται λέξη με λέξη. Παράλληλα, όταν μια λέξη εμφανίζεται πρώτη φορά στο κείμενο ανανεώνεται και ένα άλλο dictionary που καταγράφει σε πόσα κείμενα εχει εμφανιστεί η κάθε λέξη. Όταν δέχεται querry, το επεξεργάζεται και φτάχνει τον ανεστραμένο καταλογο του, με παρόμοιο τροπο που το έκανε στα κείμενα. Έπειτα, βρίσκει το tf/idf cosine similarity για κάθε url και το αποθηκεύει σε ένα dictionary (όταν δεν είναι 0). Τέλος, ταξινομεί το dictionary με το similarity και επιστρέφει τα topk αποτελέσματα, ανάλογα τα πόσα εχουν ζητηθεί.